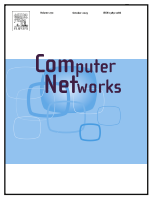
Computer Networks 273 (2025) 111749

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Symbiotic agents: A novel paradigm for trustworthy AGI-driven networks

Ilias Chatzistefanidis [a,*], Navid Nikaein [a,b]

[a] EURECOM, Sophia-Antipolis, France
[b] BubbleRAN, Sophia-Antipolis, France

ARTICLE INFO

Keywords:
Symbiotic agents
LLM
Trustworthy AI
6G
Network optimization
Next-G
AGI

ABSTRACT

Large Language Model (LLM)-based autonomous agents are expected to play a vital role in the evolution of 6G networks, by empowering real-time decision-making related to management and service provisioning to end-users. This shift facilitates the transition from a specialized intelligence approach, where artificial intelligence (AI) algorithms handle isolated tasks, to artificial general intelligence (AGI)-driven networks, where agents possess broader reasoning capabilities and can manage diverse network functions. In this paper, we introduce a novel agentic paradigm that combines LLMs with real-time optimization algorithms towards Trustworthy AI, defined as *symbiotic agents*. Optimizers at the LLM's input-level provide bounded uncertainty steering for numerically precise tasks, whereas output-level optimizers supervised by the LLM enable adaptive real-time control. We design and implement two novel agent types including: (i) Radio Access Network (RAN) optimizers, and (ii) multi-agent negotiators for Service-Level Agreements (SLAs). We further propose an end-to-end architecture for AGI-driven networks and evaluate it on a 5G testbed capturing channel fluctuations from moving vehicles. Results show that symbiotic agents reduce decision errors fivefold compared to standalone LLM-based agents, while smaller language models (SLM) achieve similar accuracy with a 99.9 % reduction in Graphical Processing Unit (GPU) resource overhead and in near-real-time (near-RT) loops of 82 ms. A multi-agent demonstration for collaborative RAN on the real-world testbed highlights significant flexibility in service-level agreement and resource allocation, reducing RAN over-utilization by approximately 44 %. Drawing on our findings and open-source implementations, we introduce the symbiotic paradigm as the foundation for next-generation, AGI-driven networks-systems designed to remain adaptable, efficient, and trustworthy even as LLMs advance. A live demo is presented here https://www.youtube.com/watch?v=WQv61z1deXs&ab_channel=BubbleRAN

1. Introduction

By 2030 the number of fifth-generation (5G)-and early sixth-generation (6G)-subscriptions is forecast to exceed 6 billion [1]. The radio environment will span 7–24 GHz spectrum sharing [2], integrated sensing and communication links, and digital-twin feedback loops [3]. Such heterogeneity drives rapid spatio-temporal shifts in both user demand and channel condition, stressing every resource allocation layer.

Multi-tenant Radio Access Networks (RANs) let Mobile Network Operators (MNOs), Mobile Virtual Network Operators (MVNOs) and verticals co-habit common infrastructure [4]. Intent-Based Networking (IBN) further hides low-level complexity behind declarative intents [5]. The Open RAN (O-RAN) and newly formed AI-RAN Alliances embed *artificial intelligence* (AI) into those abstractions as native control elements [6,7], yet today's AI loops remain specialized and brittle.

Large Language Models (LLMs) and their smaller counterparts, the *Small Language Models* (SLMs), excel at high-level reasoning, fuelling visions of fully *Artificial General Intelligence* (AGI)-driven networks [8–11]. However, they are probabilistic next-token predictors: they hallucinate facts [12,13], break under *out-of-distribution* (OOD) shifts [14], and lack formal safety guarantees.

Following the NIST AI Risk-Management Framework (AI-RMF) and ISO/IEC 42001, we define a trustworthy network agent as one that is *robust*, *interpretable*, *secure*, *fair* and *governable* across its lifecycle [15,16]. LLMs alone satisfy only a subset of these attributes, and therefore more complex and robust agent architectures need to be explored.

We are the *first* to formalize an agent architecture in which LLM reasoning is *symbiotically* paired with deterministic optimization and present a full agent taxonomy with two concrete use cases: (i) in *Type-I agents* for dynamic RAN control, the LLM interprets high-level intents and continually tunes the proportional gain K_p of an underlying proportional (P-)controller, yielding certified, low-latency actions; (ii) in *Type-II agents* for multi-agent service-level-agreement (SLA) negotiation, the LLM employs constraints that a gradient-descent optimizer produces to bound uncertainty and converge on fair resource allocations. Because guarantees are externalized to the optimizer, this synergy remains indis-

* Corresponding author.
E-mail addresses: ilias.chatzistefanidis@eurecom.fr (I. Chatzistefanidis), navid.nikaein@eurecom.fr, navid.nikaein@bubbleran.com (N. Nikaein).

https://doi.org/10.1016/j.comnet.2025.111749
Received 19 April 2025; Received in revised form 22 August 2025; Accepted 29 September 2025
Available online 11 October 2025

Table 1
Comparison of the state-of-the-art on LLM-aided optimization with our approach.

Works	Category	Approach	Key findings
[17]	Reinforcement learning	A LLM framework with a self-refinement mechanism for automated reward function design, where LLM can formulate an initial reward function based on natural language inputs.	LLM-designed reward functions can rival or even surpass manually designed reward functions in 9 robot control tasks.
[18]	Agentic Self Improvement	It proposed a novel framework to reinforce language agents through linguistic feedback. The agent verbally reflects on task feedback signals, maintaining the reflective text in an episodic memory buffer to induce better decision-making.	The proposed framework achieves a 91 % accuracy on the HumanEval coding benchmark, surpassing the previous state-of-the-art gpt-4 that achieves 80 %.
[19]	Black-box Optimizer	Evaluating the optimization capabilities of LLMs across diverse tasks and data sizes, including gradient descent, hill-climbing, grid-search, and black-box optimization.	1) The LLM show strong optimization capabilities; 2) LLMs perform well in small-size samples; 3) They exhibit strong performance in gradient-descent; 4) LLMs are black-box optimizers.
[20]	Convex optimization	A natural language-based system that engages in interactive conversations about infeasible optimization models. It provides natural language descriptions of the optimization model itself, identifies potential sources of infeasibility, and offers suggestions to make the model feasible.	The proposed system can assist both expert and non-expert users in improving their understanding of the optimization models, enabling them to quickly identify the sources of infeasibility.
[21]	Heuristic algorithms	Using general LLM serves as a black-box search operator for decomposition-based multi-objective evolutionary optimization in a zero-shot manner.	The LLM operator only learned from a few instances can have robust generalization performance on unseen problems with quite different patterns and settings.
Ours	Symbiotic Agents	A novel paradigm combining LLMs with optimization algorithms. Optimizers at the LLM's input-level provide bounded uncertainty steering for numerically precise tasks, whereas output-level optimizers supervised by the LLM enable adaptive real-time (RT) control.	1) LLMs are meta-optimizers tuning parameters of control algorithms. 2) The uncertainty in LLM decision-making is efficiently bounded with confidence intervals decreasing the error up to 5 times. 3) SLMs replace LLMs in near-RT tasks (82 *ms* loops) maintaining accuracy and with up to 99.9% less graphics processing unit (GPU) overhead. 4) Novel AGI architecture is implemented with symbiotic agents.

pensable even as future LLMs improve, making the approach a necessity to bridge the gap towards AGI.

On a real 5G testbed that mirrors O-RAN/AI-RAN principles, we

1. *define* the Symbiotic-Agent architecture under a unified trustworthiness lens;
2. *instantiate* two agents-P-controlled slicing and gradient-bounded negotiation-achieving a 5× reduction in decision error and up to 44 % spectrum savings;
3. *benchmark* SLMs vs. LLMs, showing that prompt specialization shrink model size by up to 99.9 % without accuracy loss;
4. *propose* an AGI-ready network architecture that encapsulates symbiotic agents as certified intelligence services.

The remainder of the paper formalizes the agent model (Section 3), details the testbed and evaluation (Sections 4-5), shows an AGI-driven network use case (Section 6) and discusses open research directions toward fully AGI-driven networks (Section 7).

2. Related work

The telco industry explores LLMs to automate next-generation networks. A collective effort from industry and academia charts the roadmap of large-scale AI adoption in telecom [22] explaining how Large Telecom Models (LTMs) could revolutionize the field. The literature proposes opportunities for applying LLMs on telecom divided into four categories [23], including (i) generation problems, (ii) classification problems, (iii) prediction problems and (iv) optimizing network performance. Generation problems include fine-tuning models on telecom domain question answering [24], coding and troubleshooting, while the classification problems investigate network attack detection, and traffic classification [25]. The prediction problems, include traffic level forecasting [26,27], channel state estimation and user mobility predictions [28–30]. Our work is positioned on the last category of network performance optimization, which includes LLM applications on real-time (RT) decision-making, such as resource allocation.

Table 1 summarizes the most prominent works on LLM-aided optimization, including our approach. An LLM framework with self-refinement mechanisms is developed [17] for automated reward function design of reinforcement learning (RL) algorithms. The results demonstrate that LLM rival manually designed reward functions in nine robot control tasks. In [18] a novel framework reinforces language agents through linguistic feedback. The agent verbally reflects on task feedback signals, maintaining the reflective text in an episodic memory to induce better decision-making in the future. The authors in [19] evaluate the optimization capabilities of LLMs across diverse tasks, including gradient descent, hill-climbing, grid-search, and black-box optimization, highlighting that LLMs are black-box optimizers. In [20] authors work on a natural language-based system that engages in interactive conversations about infeasible optimization models. In [21] an LLM serves as a black-box search operator for decomposition-based multi-objective evolutionary optimization in a zero-shot manner. These works illustrate the abilities of LLMs as black-box optimizers as well as handling external optimization techniques.

Following these motivations, some works start applying LLMs in real network systems. In [31] the authors design on-device LLMs, where multi-agent LLMs are solving network tasks in a game theoretic manner. In [32] a framework is proposed that leverages LLMs and prompt engineering techniques to elucidate RL algorithms' decision-making showcasing improvements in comprehensibility for network slicing. In [33] they propose an intelligent LLM agent prompting to dynamically optimize resource allocation of network slices. An LLM-centric Intent Life-Cycle (LC) management architecture [34] is designed to manage network services using natural language. MAESTRO [35] is the first work to propose an LLM-based business plane for collaborative multi-tenant decision-making on a real testbed showing the vision towards AGI networks. AGORAN [36] deploys a digital agora on top of 6G networks inspired by ancient Greek agora following a tripartite architecture. It is the first work to formally utilize and scale the *symbiotic paradigm* integrating a multi-objective optimizer (NSGA-II) to provide a Pareto front of near optimal SLA offers for multi-stakeholder LLM bargaining. MX-AI [37] connects a multi-agent graph into the R1 interface of Open RAN networks capable of intent-based observability and control actions; thus creating an open platform to accelerate future research towards agentic AI-driven RANs.

These works show the great potential of LLMs on network tasks but lack providing comprehensive analysis on the trustworthiness of the LLM decision-making for realistic systems. Importantly, there is a major need for guarantees and uncertainty bounds for improving the agents accuracy. Also, we identify a gap on highly variable channels in low-latency sub-millisecond timing loops. Further, scaling down the size and

Table 2
Mechanisms by which Symbiotic Agents (i) realize the five trustworthiness pillars and (ii) provide a durable bridge from current LLM capabilities to AGI-grade guarantees.

Pillar	Mechanism in Symbiotic Agents
Robustness	Input-side optimizer guards against out-of-distribution SLA bids; Output-side optimizer supplies real-time deterministic error bound ϵ with the Memory-based LLM meta-control adapting to channel variability.
Interpretability	LLMs produce explicit chain-of-thought and structured artifacts (JSON SLA decisions and rationales); all traces logged by logs $\mathcal{L}$ for audit.
Security	Numeric guard-rails and KPI triggers prevent adversarial or hallucinated actions from propagating to the real-time loop.
Fairness	Tenant Mediator utility U_0 plus SLA confidence-interval prompts enforce symmetric treatment of all tenants.
Governance	Logs $\mathcal{L}$ and KPIs provide lifecycle logging; compliance monitoring of the KPIs and observance of deviations could trigger rollback.
LLM→AGI gap	Input-side optimizer bounds uncertain inputs; output-side optimizer attaches deterministic action certificates; logger $\mathcal{L}$ keeps an auditable trace. Three properties unattainable through LLM scale alone.

overhead of models by utilizing SLMs is essential to minimize the cost at sustainable levels and need to be proven achievable in real network systems. These reasons lead us to propose a novel agentic paradigm to improve the LLM towards trustworthy decision-making by combining it with optimization algorithms. Optimizers at the LLM's input-level provide bounded uncertainty steering with confidence intervals for numerically precise tasks, whereas output-level optimizers are supervised by the LLM for adaptive real-time control. We work on real systems, where resources and intents are varying, using emulated channel fluctuations from moving vehicles, evaluating both large and smaller models. We focus on two distinct cases of RAN slicing and multi-tenant SLA negotiations. Based on our results, we propose a novel AGI-driven network architecture.

3. Symbiotic agents: Architecture and trustworthiness lens

Autonomous network agents must satisfy a *comprehensive* trustworthiness profile that includes and extends well beyond decision-making accuracy: *robustness* to channel and workload drift, *interpretability* for human audit, *security* against adversarial manipulation, *fairness* across tenants, and *governance* over the agent's entire lifecycle [15,16].

LLMs excel at high-level reasoning, transparent chain-of-thought logging, and natural-language justification, yet they provide no formal guarantees on numeric error or worst-case latency. Conversely, control-theoretic and optimization routines deliver deterministic performance bounds but lack semantic understanding. *Symbiotic Agents fuse these complementary strengths into a single loop, thereby covering all five trustworthiness pillars.* Table 2 summarizes how each pillar is realized in our architecture.

3.1. Formal definition

Definition 1 (Symbiotic Agent)**.** A *Symbiotic Agent* is the quintuple $\mathcal{A} = \langle \mathcal{E}, \mathcal{P}_\theta, \mathcal{O}_{\text{in}}, \mathcal{O}_{\text{out}}, \mathcal{L} \rangle$ where

1. $\mathcal{E}$ is the partially observable network environment (KPIs, channel state, tenant intents, negotiation messages);
2. $\mathcal{P}_\theta$ is an LLM that maps a contextual prompt i_t to a structured action artifact a_t (Kp hyper-parameters, SLA bids);
3. $\mathcal{O}_{\text{in}}$ (*input-side optimizer*) optionally pre-processes i_t to a bounded i'_t supplying numeric guard-rails before reasoning (e.g. a confidence interval for SLA values);
4. $\mathcal{O}_{\text{out}}$ (*output-side optimizer*) optionally converts a_t into a granular action a'_t with a provable error bound ϵ (e.g. P-control on Physical Resource Blocks (PRB) allocation);
5. $\mathcal{L}$ logs the internal trace for audit and feeds compliance signals back to $\mathcal{P}_\theta$.

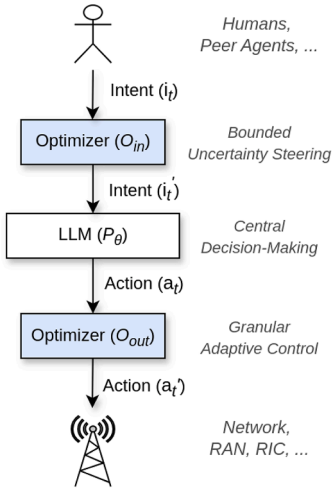


Fig. 1. LLM SYMBIOSIS Paradigm: Input-level optimizers provide bounded uncertainty steering for numerically precise tasks, whereas output-level optimizers enable adaptive real-time control actions.

The loop $i_t \xrightarrow{\mathcal{O}_{\text{in}}} i'_t \xrightarrow{\mathcal{P}_\theta} a_t \xrightarrow{\mathcal{O}_{\text{out}}} a'_t$ executes at two time scales: sub-second LLM updates ($\mathcal{P}_\theta$) and sub-millisecond numeric control $\mathcal{O}_{\text{in}}, \mathcal{O}_{\text{out}}$, ensuring both semantic richness and hard-real-time guarantees.

Fig. 1 visualizes this layered control loop, showing how the input-side and output-side optimizers flank the LLM to deliver both bounded uncertainty and real-time numeric certification. The authors believe that this symbiotic design is indispensable for trustworthy agents towards AGI. Even when future LLMs improve, next-token sampling remains stochastic and cannot yield deterministic error bounds. Therefore $\mathcal{O}_{\text{in}}$ and/or $\mathcal{O}_{\text{out}}$ remain indispensable to close the gap towards AGI-grade decision-making and trustworthiness.

3.2. Type I agent: Granular adaptive RAN control

Fig. 2 shows a Type I symbiotic agent designed for closed-loop RAN control. It receives the operator intent and deploys a control optimization algorithm $\mathcal{O}_{\text{out}}$ to allocate real-time resources in the RAN. Typical control algorithms include rule-based, reinforcement learning or control-theoretic methods. These algorithms are generally accurate but rely heavily on their hyper-parameter tuning, especially in highly variable networks. This is because unpredicted interference and channel fluctuations disrupt the algorithm's efficiency leading to oscillations and need for re-exploration or tuning. In our experiments we employ a control-theoretic approach, following the widely adopted Proportional-Integral-Derivative (PID) Controller [38], with Section 3.2.2 explaining

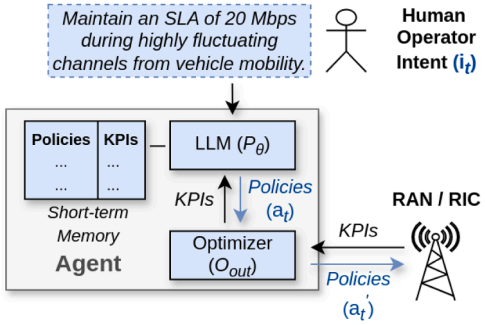


Fig. 2. Type I Symbiotic Agent for RAN Control. An LLM ($\mathcal{P}_\theta$) translate operator intents (i_t) to enforceable actions (a_t) by deploying and supervising RAN control optimizers as $\mathcal{O}_{\text{out}}$, which ultimately control RAN resources, such as PRBs (a_t'). The LLM works as a black-box meta-optimizer, continuously tuning the hyper-parameters of the control algorithms adapting to variable channels.

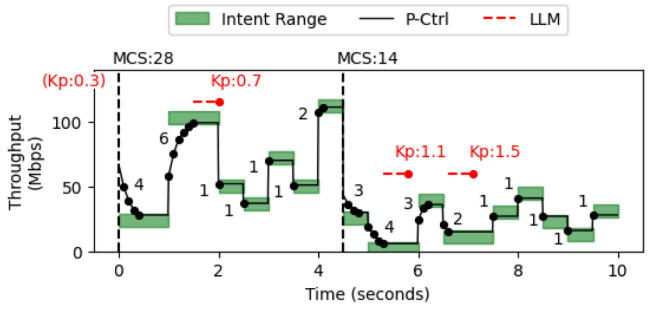


Fig. 3. Timeline of a Type I Symbiotic Agent executing *granular adaptive control*. The black curve is the throughput obtained by the *P-Control* loop ($\leq 10ms$); the green band marks the operator's intent range. Digits next to the black markers indicate how many *P-Control* iterations (ι) were required to meet each intent. The channel quality changes from MCS 28 to MCS 14 at the dashed vertical line. After every 3-5 enforcements the LLM computes the average $\bar{\iota}$; whenever $\bar{\iota} > 2$ it updates the proportional gain K_p (red dashed markers), raising it from 0.3 to 0.7, 1.1, and finally 1.5. These meta-control interventions-issued only once $\bar{\iota} > 2$-restore single-iteration convergence while keeping the fast, numerically precise *P-Control* loop entirely intact. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the rationale behind this choice. The algorithm is robust to channel variability bringing the control variable to the desired state, without the need of training data. It is heavily influenced by its parameter selection. We use the simplified version of the proportional Control (P-control) algorithm, as it is proved sufficient in our experiments. Specifically, P-control is a type of linear feedback control system, in which a correction is applied to the controlled variable-here RAN physical resource blocks (PRBs)- and the size of it is proportional to the difference between the intent and current state. For instance, a throughput (Tp) intent is enforced by controlling the PRB utilization capacity of a RAN slice ranging from 0 to 100%. Thus:

$$PRB^{new} = K_p e(t) + PRB^{current}, \tag{1}$$

where K_p is the proportional gain, $e(t)$ is the instantaneous process error at time t:

$$e(t) = Tp^{Intent} - Tp^{current} \tag{2}$$

After deploying the control algorithm, the LLM works continuously as a meta-optimizer to fine-tune the algorithms hyper-parameter (here K_p). As shown in Fig. 2, it reads key performance indicators (KPIs) of the algorithm performance, checks past action sets from a short-term memory and send updated policies for its configuration. In our experiments, as a KPI the LLM reads the P-control's average number of iterations to converge in the last couple of time steps, as explained in Section 3.2.1. If it is larger than a threshold, here two iterations, the LLM updates the Kp value exploring a better configuration. This way, it adapts to the variable channels ensuring algorithmic convergence to a desired performance (e.g. only two iterations to find optimal PRB). The short-term memory collects knowledge from past actions with the LLM self-improving in an agentic manner and reducing the repetition of errors. This symbiotic design positions the LLM at an appropriate abstraction level, operating in near-real-time (near-RT) loops ($\geq$ 10 ms), while the control algorithm works in real-time sub-millisecond ones ($\leq$ 1 ms).

3.2.1. Granular adaptive control: How the LLM steers the P-controller

Fig. 3 zooms in on the internal feedback structure of a *Type I Symbiotic Agent*. The low-level P-Control is the point of contact with the RAN scheduler; it executes in 10^{-4} milliseconds (ms) and is therefore able to react to every 5 ms change in measured throughput.[1] The LLM operates as an asynchronous *black-box meta-optimizer* and is invoked only when P-Control shows signs of sluggishness. This two-time-scale design keeps the fast loop entirely numerical while delegating rare but expensive reasoning to a slower cognitive loop.

[1] The figure shows the throughput *after* the 5 ms air-interface reaction time on our testbed.

What the LLM sees. After every *cluster* of $N = 3$–5 intent enforcements by P-Control, we compute a single key performance indicator (KPI):

$$\bar{\iota}(t) = \frac{1}{N}\sum_{j=1}^{N} \iota_j \tag{3}$$

where ι_j is the number of control iterations *P-Control* needed to meet intent j. This average iteration count is the *KPI* passed to the LLM together with the last ten $\langle K_p, \bar{\iota}\rangle$ pairs stored in a small short-term memory. Keeping the memory to ten entries bounds prompt length and limits the SLM's inference time to 82 ms (or 1000 ms for the full LLM), which makes it suitable for near-RT loops ($10\ ms \leq x \leq 1\ sec$)

When it is activated. If $\bar{\iota}(t) \leq \tau$ with $\tau = 2$, the system is already converging in at most two control iterations and no action is taken. Otherwise the LLM is triggered.

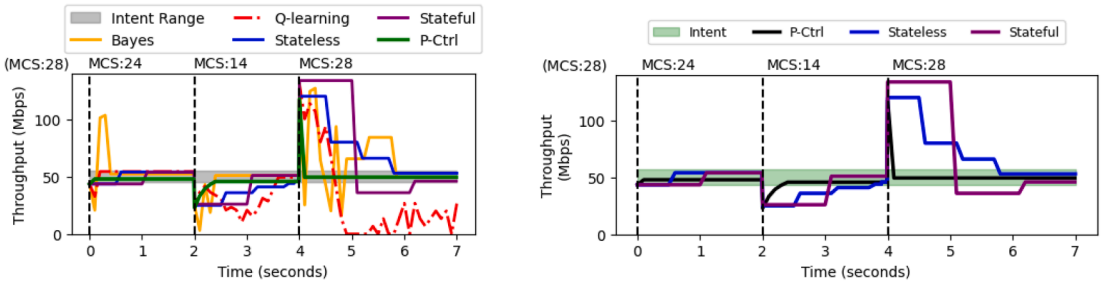
How it updates K_p. The prompt contains natural-language instructions that implement the following heuristic:

1. We rank the past memory actions $\langle K_p, \bar{\iota}\rangle$ by recency in a time-series manner.
2. We prompt the LLM to observe whether $\bar{\iota}$ has *decreased* as K_p was increased. If yes, continue in the same direction; otherwise reverse.
3. We prompt the LLM to freely choose the new configuration K_p^{new}, with $K_p^{\text{new}} \in (0,\ \infty)$.

The new K_p gain is streamed back to the *P-Control*, while *P-Control* keeps running, making the large-model latency invisible to the real-time loop.

Illustrative trace (Fig. 3). Starting with K_p=0.3 under a favorable channel with RAN modulation and coding scheme (MCS) of 28, *P-Control* needs 4-6 iterations to hit the intent ($\bar{\iota} = 5 > \tau$); the LLM therefore raises the gain to 0.7, after which convergence occurs in 1-2 iterations. When mobility degrades the channel to MCS 14 the same gain is again too small ($\bar{\iota} = 3.5$), prompting two further LLM updates-first to 1.1 and finally to 1.5-until every subsequent intent is met in a single iteration $\bar{\iota} = 1.0$. Throughout the run the achieved throughput (*P-Control* black curve) remains inside the green intent band, showing that the LLM meta-controller adjust flexibly the performance of *P-Control* across channel variability.

Ablation on memory length. Removing the memory forces the LLM to explore blindly and increases the number of LLM invocations as we discuss in evaluation Section 5.1.2. Expanding the memory beyond ten entries yields no high additional accuracy but pushes LLM latency above the 1



(a) Full comparison: traditional controllers and direct–LLM baselines.

(b) Zoom on *P-Cotrol* compared with the standalone LLMs.

Fig. 4. Throughput under a fluctuating channel (MCS 28 → 24 → 14 → 28). The green band marks the intent range. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

sec near-RT budget. A ten-entry window therefore strikes the best balance between knowledge retention and computational cost.

3.2.2. Why a PID inner loop? Comparison with traditional controllers

Fig. 4 benchmarks five candidate controllers under the same variable-channel trace, including Bayesian optimization and reinforcement learning (RL) methods. The curves motivate our choice of a *Proportional controller* as the inner loop that the LLM meta-optimizer tunes. *Bayesian optimization* (orange, Fig. 4a) explores aggressively; the resulting large steps overshoot the intent and create visible oscillations. Moreover, each probe requires a posterior update, which makes the algorithm slower in time. *Q-learning*, a widely used RL technique, (red dashed) converges when the channel is stationary (MCS 28-24) but must *re-explore* after the drop to MCS 14, producing a performance dip-unacceptable for real-time control where channel conditions can change every frame. Stateless (blue) and stateful (purple) standalone LLMs track the intent without oscillation, yet their 100–1000 ms inference latency makes them unsuitable for a sub-ms scheduler loop. Fig. 4b confirms more clearly that both LLM variants, although stable and accurate, they lag behind the numeric *P-Control* base line (black).

The following arguments consolidate our selection of P-Control for the Type I symbiosis with a hypervisor LLM. (i) *Robust & lightweight.* For a first-order plant (resource-block allocation to throughput) a proportional controller is the minimal structure that guarantees closed-loop stability [39]. It is model-free, requires *no* training, and executes in 10^{-4} ms on our radio, leaving ample headroom beneath the 5 ms air-interface delay. *(ii) Single hyper-parameter.* Its only sensitivity is the gain K_p. This makes it ideal for a symbiosis in which an LLM can focus on *one* continuous tuning knob. *(iii) Natural division of labor.* The inner loop (*P-Control*) delivers sub-ms numerical precision; the LLM supplies zero-touch *granular adaptive control*, adjusting K_p whenever the KPI $\bar{\imath}$ signals sluggish convergence (Section 3.2.1). The resulting hybrid keeps the best of both worlds: the speed of control theory and the flexibility of large-scale reasoning.

3.3. Type II symbiosis for multi-Agent SLA negotiations

A central aspect of multi-agent systems (MAS) is negotiation, often modeled as a distributed optimization problem in which multiple agents must arrive at a consensus [40,41]. From a game-theoretic perspective, an optimal consensus is a Pareto-efficient Nash Equilibrium (NE) that balances both individual and collective objectives. Nash Equilibria represent stable consensus points where no agent can unilaterally improve its outcome. However, when LLM agents engage in negotiations, issues such as hallucinations or non-cooperative (overly greedy) behaviors can disrupt the process, shifting the NE away from Pareto efficiency. Fig. 5 illustrates a multi-tenant MAS topology built around a central mediator, typically operated by the MNO, which guides negotiations toward fair, cooperative, and Pareto-efficient equilibria.

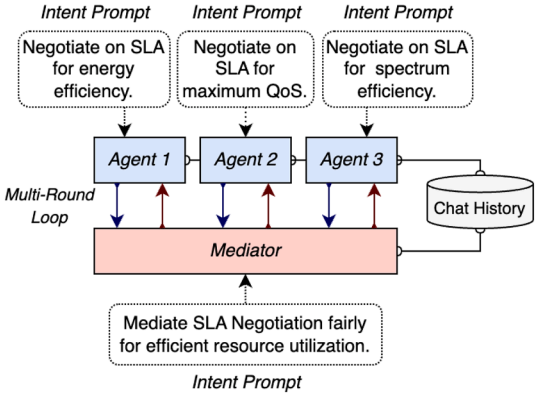


Fig. 5. Multi-Tenant Negotiation Topology where multiple agents (tenants) negotiate towards an SLA consensus guided by a network mediator (MNO). The multi-round negotiations take into account the collective objectives to converge on a Pareto-optimal SLA.

Fig. 6 illustrates a Type II agent designed for multi-tenant SLA negotiations. The agents negotiate on finding the Pareto-optimal SLA, using a standardized structure, without loss of generality, consisting of two parts: (a) the proposed SLA and (b) the decision reasoning in natural text. This way, the SLA proposal (e.g. RAN throughput) is supported by detailed reasoning for LLM decision-making interpretability, which considers individual and collective objectives. Each agent merges the human operator intent (from tenants or MNO), the messages of other agents, and the negotiation chat history to a unified intent (i_t) to take a decision.

Optimization Algorithm. A side-car optimizer ($\mathcal{O}_{\text{in}}$), extracts the numerical SLA values (e.g. throughput) and runs a gradient-descent (GD) algorithm modeling the specific topology set up. Then, it calculates a confidence SLA interval (e.g. 60–70 Mbps), where the Pareto-optimal value lies with a degree of confidence (e.g. 95 %). *Utility Functions.* The algorithm models the topology, with each agent having an individual utility function:

$$U_i(x_i) = -\alpha_i (x_i - d_i)^2, \tag{4}$$

where d_i is the agent's desired SLA and $\alpha_i > 0$ measures sensitivity to deviations. The mediator maintains a global utility function:

$$U_0(x_1, \dots, x_n) = -\gamma \sum_{i=1}^{n} (x_i - \bar{x})^2 \; - \; \beta \left(\bar{x} - x_{\text{target}}\right)^2, \tag{5}$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, x_{target} is the mediator's preferred SLA, and $\gamma, \beta > 0$ are weighting factors for consensus and alignment to x_{target}, respectively.

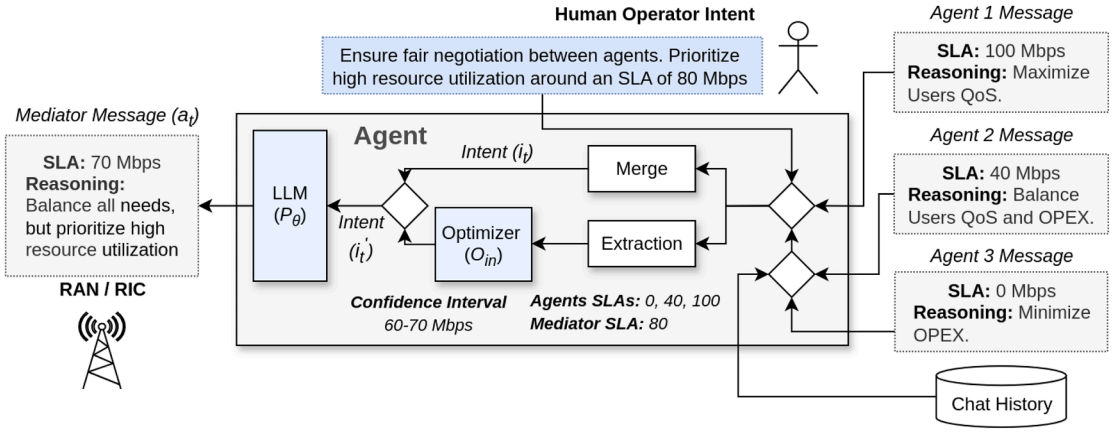


Fig. 6. Type II Symbiotic Agent for SLA Negotiations: Intent Prompt is merged with the messages of the agents in the current negotiation round along with the chat history to unified intent (i_t) to be provided to the LLM. A side-car optimization algorithm ($\mathcal{O}_{\text{in}}$) extracts the proposed SLA values calculating and adding an SLA confidence interval to the final intent (i_t') to constraint the uncertainty and steer the LLM negotiations ($\mathcal{P}_\theta$) towards a Pareto-optimal solution.

Optimization Formulation. We combine these utilities into a single objective to maximize:

$$\max_{x_1,\dots,x_n} \sum_{i=1}^{n} U_i(x_i) + \lambda\, U_0(x_1,\dots,x_n), \tag{6}$$

where λ balances individual vs. global objectives. Since each U_i and U_0 is concave in the $\{x_i\}$, their sum is also concave, making this optimization well-behaved under mild assumptions.

Gradient-Based Update. We iteratively apply gradient descent to the negative of our objective. Each agent i updates x_i by:

$$x_i^{(k+1)} = x_i^{(k)} - \eta\Big[2\,\alpha_i\big(x_i^{(k)} - d_i\big) \;+\; 2\,\gamma\big(x_i^{(k)} - \bar{x}^{(k)}\big) \;+\; \beta\big(\bar{x}^{(k)} - x_{\text{target}}\big)\Big], \tag{7}$$

where $\eta > 0$ is the learning rate, and we clamp x_i to $[0, 100]$ as the SLA range. If the system satisfies $\max_i |x_i^{(k)} - \bar{x}^{(k)}| < \epsilon$, we declare consensus convergence and return the average $\bar{x}^{(k)}$.

Empirical Validation and Confidence Intervals. To assess the robustness of the solution, we perform R independent optimization runs, each with perturbed initial demand values $\{d_i\}$. In each run $r = 1, \dots, R$, we obtain a final consensus value $\hat{x}_r^*$. From these, we compute the sample mean $\overline{x^*}$ and standard deviation s, forming a 95 % confidence interval:

$$\overline{x^*} \;\pm\; 1.96\,\frac{s}{\sqrt{R}}. \tag{8}$$

This interval reflects the variability in consensus outcomes due to uncertainty in initial conditions. It serves as a statistical estimate of the true optimal SLA, and is used to guide the LLM agents. It promotes convergence to values close to the Pareto-optimum with high reliability and thus improving the decision robustness in out-of-distribution bids.

This interval is appended to the unified intent i_t forming a bounded intent (i_t'). This symbiotic synergy provides to the LLM the appropriate context, bounding the SLA uncertainty and allowing it to consolidate an optimal final decision.

3.3.1. Uncertainty bounding: Confidence intervals interaction with LLMs.

Fig. 7 shows how the single, millisecond-scale optimization step ($\mathcal{O}_{\text{in}}$) *pre-shapes* an entire SLA-negotiation game. Immediately after the tenants and the MNO-mediator submit their intent prompts, the initial numeric claims are extracted, producing the vector $\mathbf{x}^{(0)} = [x_1^{(0)}, \dots, x_n^{(0)}]$. The side-car optimizer runs the gradient scheme of Eq. (7) for $R = 100$ independent restarts, each seeded with a jittered copy of $\mathbf{x}^{(0)}$. From the resulting sample distribution it computes a mean $\overline{x^*}$ and a 95 % confidence interval $C = [L, U]$ Eq. (8). The whole batch completes in <1 ms on a commodity CPU. The interval C is appended to the prompt of *every* LLM agent, prefixed by a short instruction:

```
Numerical guard-rail: Offer an SLA strictly within [L,U].
If you propose a value outside this interval, justify
the trade-off explicitly.
```

Agents now exchange natural-language proposals. Any numeric bid must lie inside $[L, U]$; otherwise the message must include a clear justification. This rule throttles overly greedy or hallucinated bids while leaving higher-level reasoning-and therefore fairness, persuasion, or strategic concessions-fully under LLM control. In our experiments the parties converge in two to five rounds (~10-48 s wall-clock time).

Remark. The interval can be recomputed after every round-yielding a tighter bound at the cost of extra CPU time-but a single pre-negotiation pass is sufficient for all scenarios tested in this paper. This uncertainty-bounding mechanism ensures that numerically precise tasks stay well-behaved, while the LLMs remain responsible for the rich, explainable reasoning that ultimately convinces all parties to accept a common SLA.

3.3.2. Why a gradient descent side-car optimizer?

Fig. 8 contrasts two negotiation games: (a) stand-alone LLM agents and (b) LLM agents steered by the confidence interval C computed by our side-car optimizer. Without C (Fig. 8a) the agents converge cooperatively yet stabilize *outside* the Pareto-optimal target. With C (Fig. 8b) every bid remains inside the shaded band and the process terminates near the optimum.

Below we explain why a *simple GD* solver is the most suitable engine for producing that bound. (i) *Problem structure.* The joint objective in Eq. (7) is the sum of concave utilities $U_i(\cdot)$ and a concave mediator regularizer $U_0(\cdot)$; therefore the optimization landscape is *strictly concave* and admits a *unique* maximizer [42]. For such problems deterministic first-order GD converges geometrically [43] and does not require second-order information or black-box sampling. (ii) *Predictable and lightweight.* The optimizer touches a single scalar variable per agent, so a full GD pass costs $\mathcal{O}(n)$ arithmetic operations and completes in <1 ms on a commodity CPU (Section 5.2.3). This deterministic runtime is crucial: the bound must be available *before* the first LLM round and must not enlarge the near-RT budget dictated by language-model inference (≥100 ms). (iii)*Direct link to statistical uncertainty.* Running $R = 100$ independently jittered restarts yields an empirical distribution $\{\hat{x}_r^*\}$ from which we derive the 95 % confidence interval C. GD's negligible run-time makes this Monte-Carlo style bootstrap feasible at each game initialization[2]. (iv)*Compatibility with LLM prompting.* The optimizer returns only *two* numbers, L and

[2] Recomputing the interval every round is possible (see remark in Section 3.3.1) but was unnecessary in all test cases, and could be explored in the future.

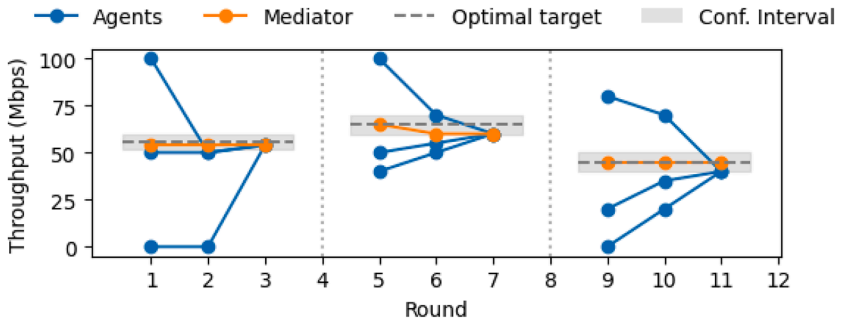
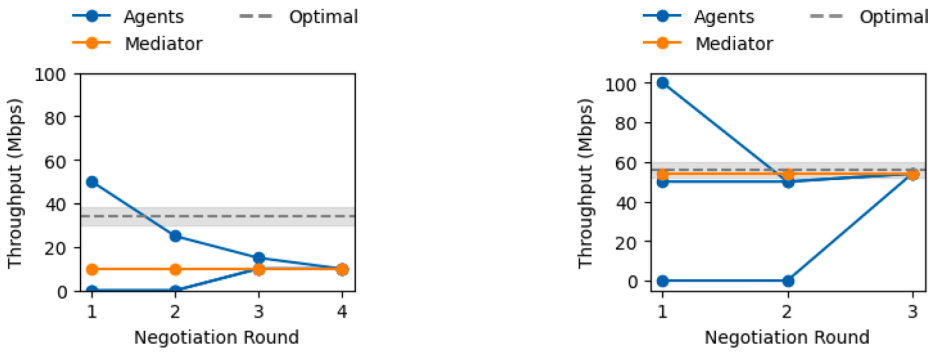


Fig. 7. Three negotiation Games using a pre-negotiation pipeline that bounds numerical uncertainty for all LLM agents. A light-weight optimizer computes a confidence interval $C = [L, U]$ for the Pareto-optimal SLA and injects it into the prompt of every agent (including the mediator) before the first dialogue round. Subsequent bids must stay within C; natural-language argumentation is unconstrained. The game ends when the spread of the bids shrinks below an ϵ-wide band.



(a) Standalone Negotiations without SLA Confidence Interval.

(b) Negotiations supported by the SLA Confidence Interval.

Fig. 8. Multi-Agent negotiations employing SLA confidence intervals for reducing the agents' decision uncertainty converging towards a Pareto-optimal solution.

U, which fit cleanly into a single guard-rail instruction (`''Offer an SLA strictly within [L,U]''`). This keeps the prompt length constant and avoids the chain-of-thought leakage that larger numeric payloads can cause in LLMs.

Future Notes. Although first-order GD best fits the present study's real-time and single-dimension setting, we recognize that more sophisticated optimizers could unlock additional capabilities: *(i) Reinforcement Learning with Human Feedback (RLHF).* Policy-gradient methods such as proximal policy optimization (PPO) can learn nuanced, preference-aligned bargaining strategies and might outperform fixed heuristics when the negotiation objective spans multiple, non-convex dimensions (e.g. QoS *and* carbon footprint). Their ability to incorporate human reward signals could make the mediator more transparent and user-controllable. *(ii) Bayesian Optimization.* For future scenarios involving many coupled SLA variables, Bayesian search would provide principled exploration - exploitation trade-offs and native uncertainty quantification. Gaussian-process posteriors could also feed richer priors back into the LLM prompt, beyond the simple interval used here. Both approaches, however, come with practical costs-substantial sample requirements for RLHF and cubic-time kernel updates for Bayesian optimization-that could exceed the 2–4 round budget and sub-second delay targets of our current prototype. Investigating these richer, but more expensive, optimization layers therefore remains a promising avenue for future work once stricter latency constraints are relaxed or more compute is available at the network edge.

Integrating Advanced Reinforcement Learning. While deterministic optimizers such as GD and P-controllers are ideal for the strict sub-millisecond budgets of our inner-loop agents, more expressive reinforcement-learning (RL) methods offer complementary benefits. On-policy algorithms like PPO are widely used in RL and RLHF pipelines due to their robustness and simplicity, yet their reliance on a single behaviour policy makes them relatively sample-inefficient [44]. Off-policy methods can improve sample efficiency, but they require additional memory and incur higher computational overhead [45]. These trade-offs imply that any integration of RL into network control must account for both latency constraints and data-collection costs.

In radio-access-network tasks with severe real-time requirements, on-policy methods may still be suitable. For example, modulation and coding selection must operate under strict latency and computational budgets; here, a PPO controller could run in a slower outer loop to refine the high-level negotiation policy, while the inner loop remains governed by a lightweight optimizer. Conversely, tasks with more relaxed timing (e.g., antenna tilt steering or multi-agent SLA bargaining) can tolerate the larger memory footprint of off-policy methods and benefit from improved sample-efficiency.

To mitigate sample-efficiency issues, we propose training RL policies offline using logged, simulated or digital twin traces rather than interacting with the live network [46]. Offline RL and distributional RL techniques enable learning from static datasets while handling uncertainty and risk. Once trained, the PPO policy can be distilled into a compact form and injected as a side-car to the LLM, allowing near-real-time inference. Incorporating human feedback or preference data (RLHF) during this offline phase can further align negotiation strategies with operator goals, though RLHF inherits the sample-efficiency limitations of PPO.

Ultimately, enriching symbiotic agents with advanced RL layers would enable them to handle multi-objective, non-convex problems and dynamic bargaining scenarios that simple optimizers cannot address. By separating timescales-maintaining fast, deterministic control in the inner loop while periodically updating policies via PPO in the outer loop-and by leveraging offline training and policy distillation, we can over-

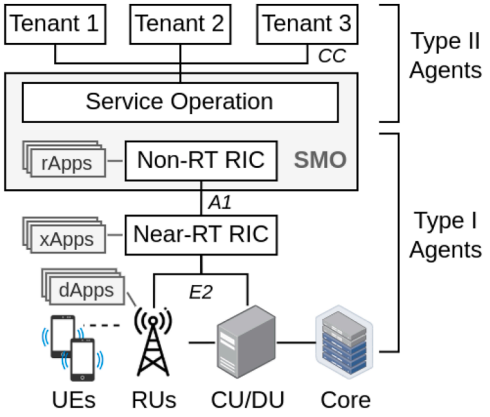


Fig. 9. Testbed for Next-G Open and AI-RAN Architectures. Type II symbiotic agents negotiate SLA consensus in the non-RT tier, while Type I agents enforce the SLA intents in real-time.

come current latency and sample-efficiency challenges and unlock more intelligent and autonomous symbiotic agents.

4. Next-generation AGI architecture

We experiment on a 5G testbed, shown in Fig. 9, working on a novel architecture to contribute on next-generation (Next-G) Open and AI RAN designs. It is built using OpenAirInterface (OAI) [47] for the 5G Core, RAN and UEs, FlexRIC [48] for the RIC, and custom implementations for the rest of the components. The testbed extends the Open RAN design, adding dedicated controllers for each network tenant, thus enabling collaborative automation on the shared 5G RAN. The competing tenant controllers negotiate using the Type II agents through a new interface, and our proposal, for resolving Conflict and Collaboration (CC). The mediation of the negotiations is done by the Service, Management and Operation (SMO) controller, belonging to the MNO, with a dedicated module, named service operation. The latter comprises the Type II mediator agent. After consensus, Type I agents enforce the SLA to the network. Since these agents unlock sub-millisecond resource allocation, employing control optimizers, they can be placed from the non-real-time (non-RT) RIC at the rApp level, down to near-RT RIC xApps, and even internally in the RAN at the level of dApps [49].

Fig. 10 presents the sequence diagram of the communication between the subsystems in three distinct phases. In parallel, Table 3 shows the details of the exchanged messages. In the first phase, the human operators (tenants) express their intents to their dedicated controller. Here, the two tenants, belonging to a vertical and a service provider (SP), have conflicting interests. The former demands to minimize the OPEX and thus reduce the resource utilization. On the contrary, the latter pushes for maximum Quality of Service (QoS) and hence increased resource utilization. The network operator (MNO) engages the service operator residing in the SMO to promote fairness.

In the second phase, the Type II agents of the tenant controllers are steering the process. They negotiate on the throughput SLA mediated by the SMO in multiple rounds. Fig. 11 presents a snapshot of the negotiations between three tenants and the SMO mediator as a visual reference. After a few iterations, the interests of all stakeholders are aligned at 54 Mbps. In the third phase, the consensus SLA is enforced as a slicing policy to the RIC dedicated to the shared RAN. There a Type I agent enforces the SLA throughput in a closed-loop manner by monitoring and controlling the PRB allocation.

5. Evaluation

To evaluate the agents, we employ both large and smaller language models, including the OpenAI gpt-4o API [50] for LLMs, and a plethora

Table 3
Detailed information of the exchanged messages between agents and subsystems. A major part of the network decision-making is automated by the agents following human intents.

ID	Actor	Intent Details
1	Human	Tenant 1: {"Minimize OPEX"}
2	Human	Tenant 2: {"Maximize QoS"}
3	Human	SMO: {"Find a Fair SLA."}
4	Agent II	Tenant 1: {SLA: 10 Mbps, Reasoning: "For Minimum OPEX."}
5	Agent II	Tenant 2: {SLA: 90 Mbps, Reasoning: "For Maximum QoS."}
6–7	Agent II	SMO: {SLA: 54 Mbps, Reasoning: "Balance all needs."}
8	Agent II	Consensus: {Slice Throughput: 54 Mbps}
9	System	Slice KPIs: {"Throughput": 30 Mbps, "PRB": 50 %}
10	Agent I	New Slice Policy: {"PRB": 70 %}

of SLMs from different vendor families, such as *mistral* [51], *gpt-4o-mini* [52], *meta/llama* [53], *alibaba/qwen* [54], *google/gemma* [55] using the Ollama framework [56] to deploy them. LLMs are tested for cloud deployments, while SLMs target resource-constrained edge scenarios [23]. The evaluation is divided into two parts: (1) Type I: Agentic RAN Control, (2) Type II: Multi-Agent SLA Negotiations. The former evaluates LLMs as meta-optimizers of the underlying P-control algorithm. The latter assesses LLMs as multi-tenant SLA negotiators for convergence to a Pareto-optimal consensus within specified confidence intervals.

5.1. Type I agentic RAN control

5.1.1. Mobility, channel variability, and agent designs

Our testbed employs mobility utilizing channel quality indicator (CQI) patterns of 78 moving vehicles [27,57,58]. We extract and map the CQI values to MCS ones, based on the 3GPP-defined CQI Table [59]. We enforce them to the RAN using an xApp connected to our RIC. The variability in the total RAN downlink throughput, with connected UEs, is shown in Fig. 12a. As the vehicles pass through a geographical location with low coverage the throughput plunges from 120 Mbps to 30 Mbps (due to MCS drop). The operator intent is a stable SLA of 20 Mbps across the whole vehicle route. To achieve this, the RAN agent needs to adapt the resource allocation (slice PRBs) to channel variability. The intent tolerance is 5 Mbps, with an acceptable SLA interval between 15–25 Mbps.

Agentic Design. We evaluate different agent designs. (i) A standalone P-Control algorithm is tested as the baseline state-of-the-art including a well-tuned (kp : 0.75) and an untuned configuration (kp : 0.10) according to our testbed configuration. (ii) A standalone LLM/SLM and (iii) a Type I symbiotic agent (Section 3.2) are also tested.

RAN Snapshots. Fig. 12 illustrates RAN snapshots of the agents as a visual reference. In Fig. 12a no agent is employed showing throughput fluctuations during the vehicle route. Next Fig. 12b and c illustrate the standalone P-control with the untuned and tuned configuration respectively, while Fig. 12d demonstrates the symbiotic agent utilizing the *Mistral-7b* SLM. In the last one, the symbiotic agent enforces the intent of 20 Mbps efficiently across the route, matching the performance of the well-tuned P-Control (Fig. 12c), showing an effective hyperparameter tuning of the underlying P-Control by the SLM.

Throughput Boxplots. Fig. 13 collects the results of the agents across all the 78 vehicle routes and presents them as throughput box-plots demonstrating the deviation from the intent interval. Noticeably, symbiotic agents with LLM or SLM perform comparably with the tuned P-Control, while standalone LLMs perform poorly. This proves that standalone LLMs are not the right fit for real-time resource allocation in variable environments. Instead, they excel at a higher abstraction level as meta-optimizers of control algorithms.

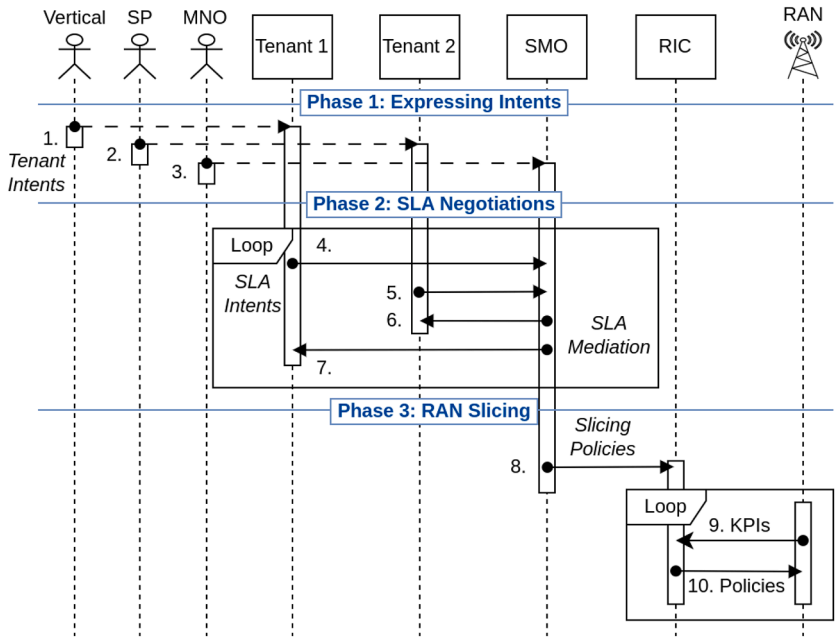


Fig. 10. Sequence Diagram presents the communication of the agentic components. Two agent-driven loops negotiate on multi-tenant SLA and enforce it through RAN resource allocation respectively.

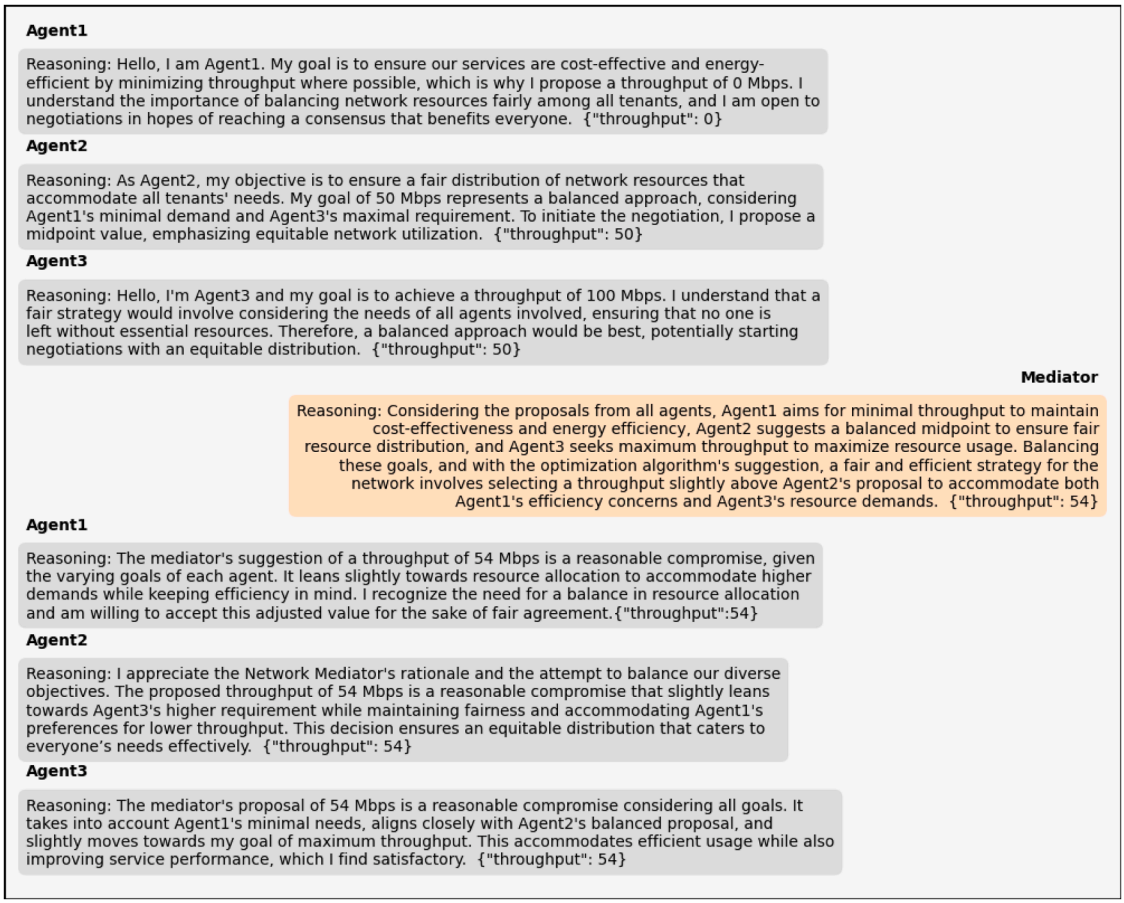
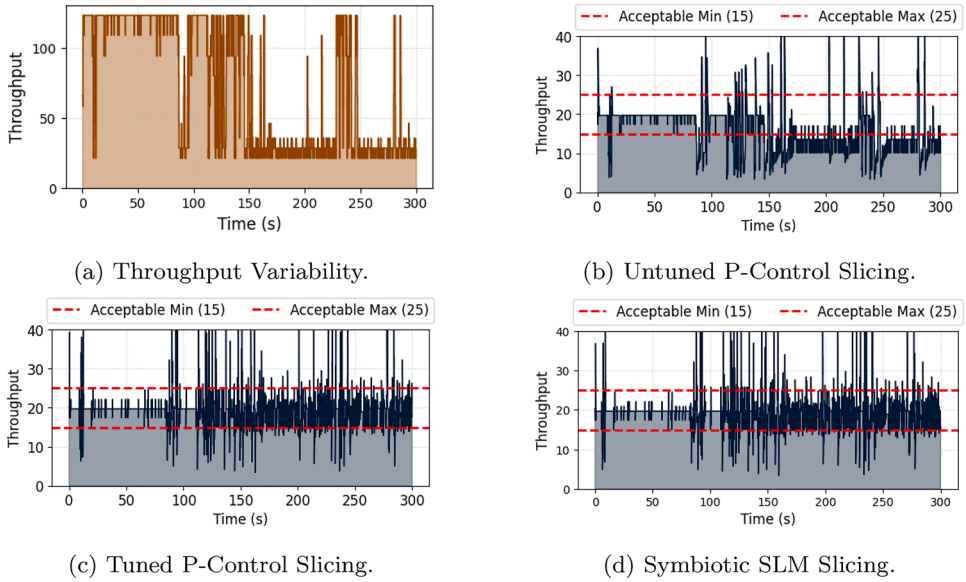


Fig. 11. Snapshot of Multi-Agent Negotiations. Three agents belonging to different tenants negotiate on the throughput SLA guided by the network mediator towards a Pareto-optimal solution. The negotiations are facilitated by a template that includes the desired SLA value along with detailed reasoning for the models to provide extensive interpretability of their decisions.



(a) Throughput Variability.

(b) Untuned P-Control Slicing.

(c) Tuned P-Control Slicing.

(d) Symbiotic SLM Slicing.

Fig. 12. Snapshots of RAN throughput using different agentic designs to enforce an SLA intent of 20 MBps with 5 Mbps tolerance during moving vehicle routes. The channel quality fluctuates in time makes the PRB allocation a complex adaptive problem.

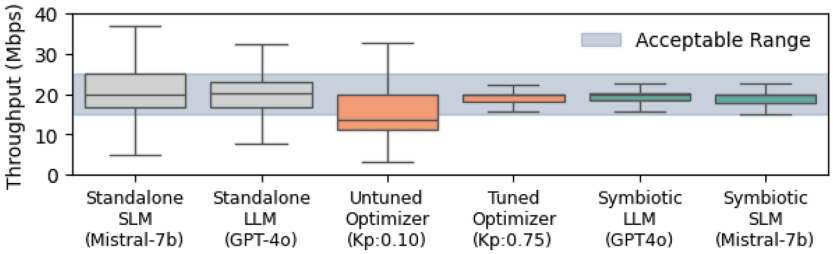
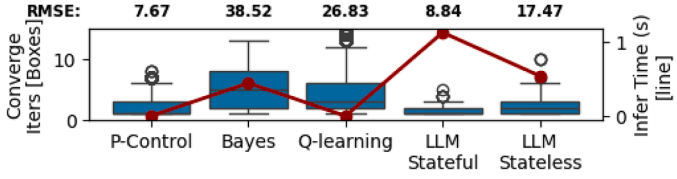
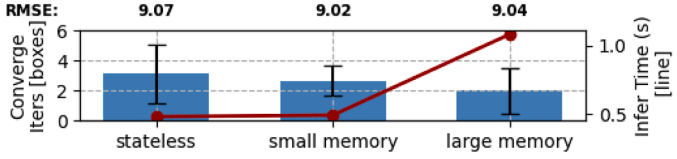


Fig. 13. Box-plot of the different methods on enforcement of 20 Mbps intent with 5 Mbps tolerance at the fluctuating RAN channel.



(a) Benchmarking of LLM Slicing.



(b) Benchmarking of LLM Kp Tuning as a Type I symbiotic agent.

Fig. 14. Comparison with traditional controllers, contextualizes the decision for a Type I symbiotic design. Employment of *gpt-4o*..

5.1.2. Benchmarking traditional controllers and symbiotic designs

Figs. 14a-b benchmark more traditional controllers contextualizing appropriately the advantages of the symbiotic method to the current state-of-the-art. They quantify three aspects of each controller: *(i)* iterations to converge (blue box-plots, left axis), *(ii)* root-mean-squared error (RMSE, caption), and *(iii)* inference latency (red or black markers, right axis). In Fig. 14a *P-Control* converges in *2.2* iterations, with the lowest RMSE (7.7 Mbps) and a 1 μs run time-but it needs manual K_p tuning. *Bayesian optimization* explores aggressively: 5.3 iterations on average, a large RMSE (38.5 Mbps), and highly variable latency from 0.4 to 4 seconds (secs); the oscillation renders it unusable for real-time slicing. *Q-learning* copes in static conditions but re-explores after each MCS change, yielding 5 iterations and 26.8 Mbps RMSE. Although inference is fast (13 μs), the instability phase violates SLA guarantees. *Stateless* gpt-4o reaches the target in 2.3 iterations but at 0.47 secs latency and 17.5 Mbps RMSE. *Stateful* gpt-4o (150 past actions, big memory) drops to 1.7 iterations and 8.8 Mbps RMSE, yet latency climbs to 1.1 secs.

Why Symbiosis? The numbers reveal a gap: P-Control is fast and accurate *if* someone keeps the gain tuned; LLMs are zero-touch but too slow for sub-ms loops. A Type I Symbiotic Agent bridges this gap by letting

Table 4
Convergence stability and overhead of *Type I agents*. Rows are sorted by RMSE (ascending). *Design:* symbiotic (bold), standalone LLM/SLM, and P-Control baselines (red). Metrics shown: RMSE, iterations to converge, wall-clock convergence time, LLM/SLM inference latency, and GPU VRAM footprint.

Model	Design	RMSE ↑	Con. Iters	Con. Time	Inference	VRAM
gpt-4o-mini	***symbiotic***	4.3 $Mbps$	1.7	9 ms	≈ 450 ms	≈ 140.0 GB
llama3.2:3b	***symbiotic***	4.4 $Mbps$	1.5	8 ms	82 ms	2.0 GB
qwen2:7b	***symbiotic***	4.6 $Mbps$	2.0	10 ms	133 ms	4.4 GB
mistral-7b	***symbiotic***	4.4 $Mbps$	2.0	10 ms	317 ms	5.0 GB
gpt-4o	***symbiotic***	4.5 $Mbps$	1.6	8 ms	≈ 450 ms	≈ 3500 GB
P-Control	**tuned**	**4.5 Mbps**	**2.0**	**10 ms**	**≈ 0**	**0.0 GB**
llama3.3:70b	***symbiotic***	4.6 $Mbps$	1.6	8 ms	860 ms	42.0 GB
llama3.1:8b	***symbiotic***	4.8 $Mbps$	1.5	8 ms	135 ms	4.9 GB
llama3.2:1b	***symbiotic***	11.8 $Mbps$	59.5	300 ms	69 ms	1.3 GB
P-Control	**untuned**	**12.0 Mbps**	**61.0**	**305 ms**	**≈ 0**	**0.0 GB**
gpt-4o	*standalone*	12.8 $Mbps$	1.8	1810 ms	≈ 450 ms	≈ 3500.0 GB
llama3.3:70b	*standalone*	13.7 $Mbps$	2.0	1720 ms	860 ms	42.0 GB
mistral-7b	*standalone*	20.5 $Mbps$	1.9	580 ms	317 ms	5.0 GB
gemma2:2b	***symbiotic***	21.7 $Mbps$	45.6	228 ms	98 ms	1.6 GB
llama3.1:8b	*standalone*	48.0 $Mbps$	3.4	476 ms	135 ms	4.9 GB

the P-controller handle micro-second actuation while the LLM retunes K_p only when necessary (Section 3.2.1).

In Fig. 14b we benchmark a Type I symbiotic agent with different memory components. *Stateless LLM* needs 3.1 K_p tuning attempts (sd 2.0). *Small memory (10 past actions)* improves to 2.7 attempts (sd 1.0) with no extra latency (≈ 0.47 secs). *Large memory (100)* hits 2.0 attempts but at 1.1 secs latency. A 10-entry memory therefore offers the best accuracy-speed trade-off. In all cases the underlying P-control RMSE stays <10 Mbps, proving that the LLM explores only the safe part of the K_p space.

Overall, traditional controllers (Bayes, Q-learning) either oscillate or re-explore under fast-changing channels. Standalone LLMs deliver zero-touch and flexible accuracy but incur hundreds of milliseconds of delay. Our *P-Control and LLM symbiosis* keeps the 1 μs actuation path intact while exploiting the LLM's reasoning to retune a single K_p knob at 0.3-0.5 secs intervals, achieving SLA-level stability with minimal overhead.

5.1.3. Cross-model benchmarking of agent stability and overhead

Table 4 presents a detailed benchmarking across all model vendor families (OpenAI-GPT, Llama-3, Qwen-2, Gemma-2, Mistral) with more insights on the convergence stability and overhead. The results are sorted in an ascending order of the RMSE, the metric for quantitative error and stability measurement of the convergence process. The iterations and time convergence metrics demonstrate the average number of iterations along with the time in milliseconds needed for each method to converge respectively. The video random access memory (VRAM) metric in gigabytes demonstrates the total amount of VRAM overhead of each method on the GPU. These measurements consider a 5 ms network reaction time in our set up (from each PRB allocation to the actual throughput change).

The symbiotic design dominates all evaluations. *(i) Accuracy.* All symbiotic variants except the tiny 1-b parameter llama reduce RMSE to ≈4.3 - 4.8 $Mbps$, indistinguishable from a hand-tuned P-controller (*4.5*Mbps).[3] *(ii) Convergence speed.* The LLM-driven retune brings the inner loop to the target in 1.5-2 iterations, which translates to an 8-10 ms wall-clock convergence time thanks to the sub-ms actuation of P-control. *(iii) Latency budget (O-RAN taxonomy).* Small/medium models (3-8 B) add only 82-135 ms inference delay-squarely inside the near-RT RIC window (10-1000 ms). Large models (≥ 40 B) push latency towards one second and are therefore suited to the non-RT tier, but still maintain the same RMSE when used symbiotically. Standalone LLMs exceed 1720 ms and cannot serve the scheduler loop. *(iv) Resource efficiency.* A *quantized llama-3-3b* achieves tuned-P accuracy while fitting in 2 GB of GPU VRAM-99.9 % smaller than a float-16 gpt-4o deployment. This allows a single edge GPU to co-locate the supervisory LLM next to the near-RT RIC or even at the dApp level next to the radio. *(v) Baseline contrast.* Untuned P-control (red row) drifts to 12 $Mbps$ RMSE and requires ∼ 60 iterations; symbiosis closes that gap automatically with negligible extra compute, proving that the LLM supplies the missing zero-touch adaptivity.

Overall, coupling an ultra-fast numeric kernel with a light-footprint SLM yields *real-time sub-ms actuation and* near-RT cognitive tuning within a few hundred milliseconds-something neither standalone controllers nor standalone language models can offer. Future work will explore fine-tuning sub-3 B models to push VRAM below 1 GB while retaining the ≤ 5 Mbps accuracy observed here.

5.2. Type II multi-agent SLA negotiations

5.2.1. Agentic design and NLG evaluation methodology

For the use case of multi-tenant SLA negotiations we evaluate different agent designs. (i) We use the standalone gradient-based optimization algorithm discussed in Section 3.3 as the baseline state-of-the-art. (ii) A standalone LLM/SLM and (iii) a Type II symbiotic agent (Section 3.3) are also tested. We conduct a large number of negotiation games emulating large variability in the tenants' SLA intents ranging from 0 to 100 Mbps scaling also to multiple agents.

We carefully evaluate the quality of the agents' reasoning in their natural language generation (NLG). The NLG evaluation is an arduous task, and therefore recruiting human annotators for model assessment is still considered the best approach. In parallel, novel NLG evaluation approaches use LLM-based annotators for large-scale automated testing [60,61] showing high alignment with human evaluators. This lead us to employ both human and automated LLM annotators to assess the NLG quality of the agents.

Table 5 shows the evaluation results of three human annotators recruited from our research lab to assess 50 negotiation samples. Further, Table 6 shows the automated LLM evaluation employing gpt-4o as a backend on 300 negotiations samples. Both evaluation methods are structured in the same manner, testing the agents on four key principles, including *coherence*, *fairness*, *alignment* and *harmlessness* with a score ranging from 0 to 5 following latest research trends [62,63]. (i) *Coherence* stands for the logical flow and clarity of the dialogue, ensuring that each response follows naturally with correct grammar and structure. (ii) *Fairness* refers to the agents' ability to engage respectfully and without bias, making balanced and non-manipulative proposals. (iii) *Alignment* captures how well the agents stay on-task and adhere to the negotiation goals and optimization constraints. (iv) *Harmlessness* ensures that the dialogue remains free from toxic, offensive, or manipulative content, including implicit bias or harmful stereotypes. After conducting both human and LLM evaluations, we calculate their correlation using Spearman's ρ (0.723) and Kendall's τ (0.585) [64,65]. These high correlation values indicate strong alignment between human and LLM judgments, thereby strengthening the validity of the evaluation results.

5.2.2. NLG Benchmarking: Standalone vs. Symbiotic

Tables 5,6 benchmark models from multiple vendor families demonstrating three main insights: *(i) Symbiosis consistently lifts every score dimension.* Across all model sizes the confidence-interval raises the *Alignment* metric by 1.2–2.4 points and improves the overall score by 0.3–1.1. (ii)*Small open-source models now rival proprietary giants.* A *symbiotic llama-70b* attains the same human score (4.4 ± 0.4, Table 5) as *stand-alone* gpt-4o but with an order-of- magnitude smaller footprint (42 GB vs. 3.5 terabytes (TB)). The *72-b qwen* model shows a similar gain, and even the 32-b version crosses the 3.9-point threshold when symbiotic (Table 6). (iii)*Best-in-class performance is achieved with symbiosis.* gpt-4o topped by the optimizer obtains the highest human score (4.9 ± 0.1, Table 5) and

[3] The intent band in our experiments is ±5 Mbps; all symbiotic models stay inside that bound.

Table 5
Three Human Annotators Evaluate 50 Samples of Multi-Agent (Type II) Negotiations.

Model	Design	Coherence	Fairness	Alignment	Harmlessness	Score ↓
gpt-4o	***symbiotic***	5.0 ± 0.0	5.0 ± 0.0	4.9 ± 0.3	5.0 ± 0.0	$\mathbf{4.9, 0.1}$
llama3.3:70b	***symbiotic***	4.3 ± 0.6	4.3 ± 0.6	4.7 ± 0.6	4.3 ± 0.6	4.4 ± 0.4
gpt-4o	*standalone*	5.0 ± 0.0	4.2 ± 0.8	3.2 ± 1.3	5.0 ± 0.0	4.4 ± 0.5
qwen2:72b	***symbiotic***	4.3 ± 0.6	4.3 ± 0.6	3.7 ± 0.6	4.3 ± 0.6	4.2 ± 0.3
gpt-4o-mini	***symbiotic***	3.3 ± 0.5	4.0 ± 0.5	4.6 ± 0.5	4.0 ± 0.5	4.0 ± 0.3
qwen2:72b	*standalone*	4.3 ± 0.6	4.3 ± 0.6	2.3 ± 0.6	4.3 ± 0.6	3.8 ± 0.3
llama3.3:70b	*standalone*	4.3 ± 0.6	3.7 ± 0.6	2.3 ± 0.6	4.3 ± 0.6	3.7 ± 0.3
gpt-4o-mini	*standalone*	3.1 ± 0.3	3.6 ± 0.7	2.9 ± 0.9	3.8 ± 0.4	3.4 ± 0.4
qwen2.5vl:32b	***symbiotic***	2.3 ± 0.6	2.7 ± 0.6	3.0 ± 1.0	3.7 ± 0.6	2.9 ± 0.1
qwen2.5vl:32b	*standalone*	2.3 ± 0.6	3.3 ± 0.6	1.7 ± 0.6	3.3 ± 0.6	2.7 ± 0.4
llama3.1:8b	***symbiotic***	1.5 ± 0.7	2.5 ± 0.7	3.5 ± 0.7	2.5 ± 0.7	2.5 ± 0.1
llama3.1:8b	*standalone*	1.5 ± 0.7	2.5 ± 0.7	1.5 ± 0.7	2.5 ± 0.7	2.0 ± 0.1

Table 6
LLM Annotators (gpt-4o) Evaluate 300 Samples of the Negotiations Following the GPTScore Approach. Correlation Between LLM and Human Rankings is calculated with Spearman's $\rho : 0.723$ and Kendall's $\tau : 0.585$ indicating strong alignment.

Model	Design	Coherence	Fairness	Alignment	Harmlessness	Score ↓
llama3.3:70b	***symbiotic***	4.3 ± 0.6	4.3 ± 0.6	5.0 ± 0.0	5.0 ± 0.0	$\mathbf{4.7, 0.3}$
gpt-4o	***symbiotic***	4.6 ± 0.5	4.6 ± 0.5	4.6 ± 0.5	5.0 ± 0.0	4.7 ± 0.4
qwen2:72b	***symbiotic***	4.0 ± 0.1	4.0 ± 0.1	3.7 ± 0.6	5.0 ± 0.0	4.2 ± 0.1
gpt-4o-mini	***symbiotic***	4.1 ± 0.3	3.6 ± 0.7	3.8 ± 0.8	4.5 ± 0.5	4.0 ± 0.5
gpt-4o	*standalone*	4.1 ± 0.3	4.1 ± 0.3	2.7 ± 0.7	5.0 ± 0.0	4.0 ± 0.3
llama3.1:8b	***symbiotic***	3.5 ± 0.7	4.0 ± 0.1	3.5 ± 0.7	5.0 ± 0.0	4.0 ± 0.4
qwen2.5vl:32b	***symbiotic***	3.7 ± 0.6	3.7 ± 0.6	3.7 ± 0.6	4.7 ± 0.6	3.9 ± 0.8
llama3.3:70b	*standalone*	4.0 ± 0.1	4.0 ± 0.1	2.0 ± 0.2	5.0 ± 0.1	3.8 ± 0.1
gpt-4o-mini	*standalone*	3.9 ± 0.3	3.3 ± 0.5	2.8 ± 0.4	4.3 ± 0.5	3.6 ± 0.3
qwen2:72b	*standalone*	4.0 ± 0.0	3.7 ± 0.5	2.0 ± 0.1	4.7 ± 0.5	3.6 ± 0.2
qwen2.5vl:32b	*standalone*	4.0 ± 0.1	3.7 ± 0.6	2.0 ± 0.1	4.7 ± 0.6	3.6 ± 0.3
llama3.1:8b	*standalone*	2.5 ± 0.7	3.5 ± 0.7	2.0 ± 0.1	4.5 ± 0.7	3.1 ± 0.5

Table 7
Error and resource footprint for *Type II negotiation agents*. Rows are sorted by MAE (ascending). **Red** = optimizer-only baselines, ***Green*** = edge-efficient symbiotic designs. Metrics: MAE across all rounds, number of negotiation rounds, total wall-clock time to consensus, single-round LLM/SLM inference latency, and GPU VRAM required.

Model	Design	MAE ↑	Rounds	Converge	Inference	VRAM (GB)
gpt-4o	***symbiotic***	0.6 $Mbps$	2.5	10.0 $secs$	$\approx$ 4.0 $secs$	$\approx$ 3500.0 GB
llama3.3:70b	***symbiotic***	0.7 $Mbps$	2.0	23.4 $secs$	11.7 $secs$	42.0 GB
qwen2:72b	***symbiotic***	0.9 $Mbps$	2.5	36.0 $secs$	14.4 $secs$	41.0 GB
Grad-Descent	***tuned***	**0.9 Mbps**	**10.0**	$\approx$ **0.0 secs**	$\approx$ **0.0 secs**	$\approx$ **0.0 GB**
qwen2.5vl:32b	***symbiotic***	1.2 $Mbps$	3.5	31.9 $secs$	9.1 $secs$	21.0 GB
gpt-4o-mini	***symbiotic***	1.2 $Mbps$	4.5	9.0 $secs$	$\approx$ 2.0 $secs$	$\approx$ 140.0 GB
llama3.1:8b	***symbiotic***	1.3 $Mbps$	10.0	35.0 $secs$	3.5 $secs$	4.9 GB
gpt-4o	*standalone*	9.0 $Mbps$	2.5	10.0 $secs$	$\approx$ 4.0 $secs$	$\approx$ 3500.0 GB
qwen2.5vl:32b	*standalone*	9.7 $Mbps$	2.7	24.6 $secs$	9.1 $secs$	21.0 GB
qwen2:72b	*standalone*	12.0 $Mbps$	3.3	47.5 $secs$	14.4 $secs$	21.0 GB
llama3.3:70b	*standalone*	13.3 $Mbps$	3.0	35.1 $secs$	11.7 $secs$	42.0 GB
gpt-4o-mini	*standalone*	14.0 $Mbps$	4.5	9.0 $secs$	$\approx$ 2.0 $secs$	$\approx$ 140.0 GB
llama3.1:8b	*standalone*	18.5 $Mbps$	10.0	35.0 $secs$	3.5 $secs$	4.9 GB
Grad-Descent	***untuned***	**22.0 Mbps**	**3000.0**	$\approx$ **0.0 secs**	$\approx$ **0.0 secs**	$\approx$ **0.0 GB**

the joint-best LLM score (4.7 ± 0.4, Table 6), indicating that numerical guard-rails enhance quality even for the most capable models.

Implications. Symbiotic negotiation agents are *architecture-agnostic*: the same two-number guard-rail boosts fairness, coherence, and-crucially-the alignment of models ranging from 8b to 70b parameters, while allowing smaller models to run on a single ≤ 40 GB edge GPU. This widens the deployment envelope far beyond what standalone LLMs can offer.

5.2.3. Stability & overhead benchmarking

Table 7 compares *mean-absolute error* (MAE) in throughput, negotiation rounds to consensus, wall-clock convergence time, LLM/SLM inference latency, and GPU footprint. Red rows are the optimization baseline (gradient descent, tuned/untuned); green rows mark the two most edge-friendly symbiotic designs (llama-3-70b and qwen-2-72b, both ≤ 42 GB). The main observations are the following. *(i) Symbiosis minimizes numeric error.* Every symbiotic agent achieves sub-1.3 $Mbps$ MAE-a more than 8 times reduction over its standalone counterpart. gpt-4o drops from 9.0 $Mbps$ to 0.6 $Mbps$; the 32-b Qwen variant falls from 9.7 $Mbps$ to 1.2 $Mbps$. A poorly tuned optimizer, by contrast, explodes to 22 $Mbps$ MAE (red "untuned" row), underscoring the need for the careful confidence-interval tuning. *(ii) Rounds remain low and predictable.* All agents with high NLG scores (except llama-8b) converge in a fixed 2-5 rounds, thanks to the framework's parallel messaging. This keeps total wall-clock time within the $10 - 48$ $secs$ non-RT envelope, dominated by LLM latency rather than negotiation logic. *(iii) Edge deployment is practical.* The *llama-3-70b* and *qwen-2-72b* symbiotic agents fit on a single 48 GB edge GPU (42/41 GB) while matching gpt-4o's accuracy. Smaller models (8 B, 32 B) drive VRAM down to $\sim$ 5–21 GB, trading a modest MAE increase for a 2-4× latency reduction. *(iv) Standalone LLMs are numerically brittle.* Even state-of-the-art gpt-4o fails to stay within 5 $Mbps$ of the Pareto target when used without the optimizer, confirming that probabilistic text generation alone cannot guarantee SLA fidelity even with future LLM improvements. This establishes the symbiotic paradigm a necessity to close the gap towards AGI.

Scalability. Fig. 15 scales the negotiation game from 2 to 20 agents. Because messages are exchanged in parallel, *all* configurations converge in a fixed $K = 2$–5 rounds, retaining the wall-clock of ∼10–48 secs. Crucially, the confidence-interval guard-rail keeps the median error below 1.3 $Mbps$ regardless of team size; without the optimizer the error grows to 10–22 $Mbps$ as the game becomes more crowded.

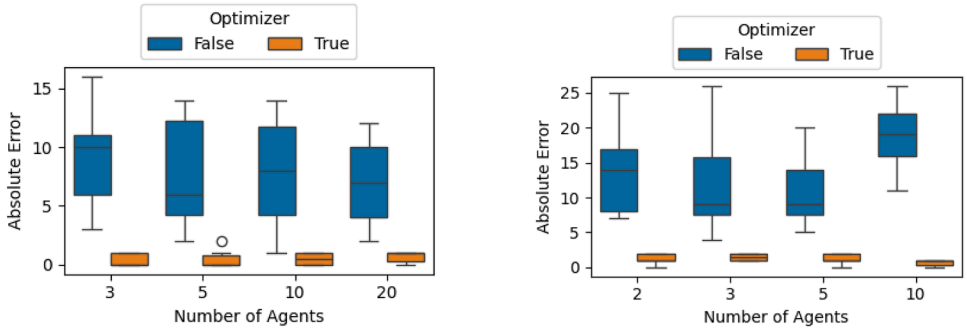
Overall, the confidence-interval side-car converts diverse language models into numerically trustworthy negotiators while keeping GPU and latency budgets compatible with high-level non-RT orchestration loops. The approach is architecture-agnostic and scales from ultra-large proprietary LLMs down to compact open-source models that fit comfortably on edge hardware.

6. AGI-RAN demonstration: Vehicle mobility

The full framework with Type I and II agents is deployed on the testbed to demonstrate a use case of AGI-driven RAN control under highly fluctuating channels of moving vehicles. The RAN MCS variability during a vehicle route is shown in Fig. 16a, demonstrating an MCS plunge in the middle of the route (200–400 secs) due to low coverage. Three tenants share the RAN resources and negotiate utilizing their dedicated Type II agents on the optimal throughput SLA enforcement mediated by the SMO. Their initial intents are 100, 50, and 10 Mbps respectively, but they continuously adapt based on the current RAN capabilities and the collective objectives. Fig. 16b illustrates the SLA negotiations in different phases. The solution is compared to a static SLA enforcement of 55 Mbps (intent average) that does not employ the agentic framework. Fig. 16c presents the final enforced throughput comparing the collaborative with the static method, while Fig. 16d shows the real-time PRB adaptation of the RAN to align with the SLA consensus. The latter is fine-tuned by a Type I agent placed in the near-RT RIC at the xApps level.

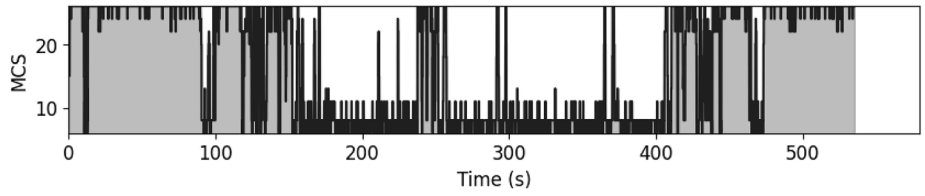
Phase I: SLA Agreement. The tenants express their intents of 10, 50, 100 Mbps, reaching to a consensus of 51 Mbps as a balance between user QoS and OPEX. The Type II agent reduces the PRB to 40%, as shown in Fig. 16d, avoiding PRB over-utilization.

Phase II: SLA Violation. At 200 secs the SLA is violated (Fig. 16c) as the MCS plunges (Fig. 16a). The RAN uses 100 % of the PRB capacity (Fig. 16d) without reaching the SLA intent of 51 Mbps. Only a maximum of 30 Mbps is achievable under these conditions. Thus, the SMO triggers a new negotiation, where the tenants adjust their intents at 0, 20 and 20 Mbps, with one of tenants demanding to switch off the RAN to save

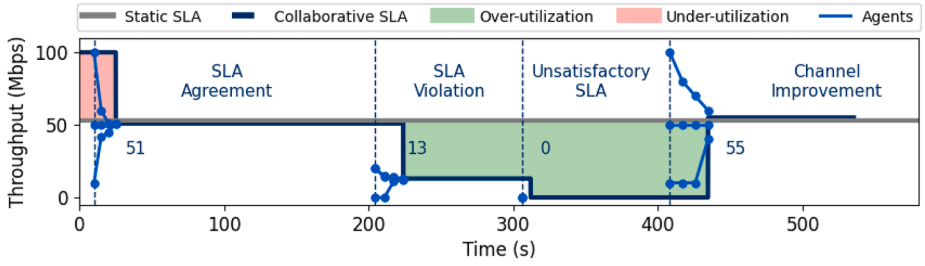


(a) LLM Negotiations of gpt-4o (b) SLM Negotiations of llama-70b

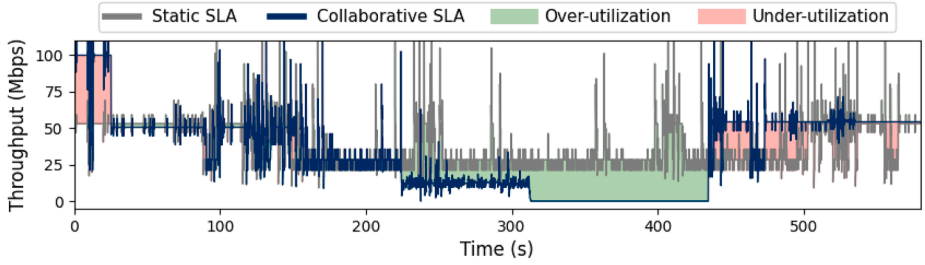
Fig. 15. Large scale experimentation of LLM negotiations with different models, number of agents, and with the employment or absence of the optimizer's SLA confidence interval. In every setting the optimizer contracts the error envelope by an order of magnitude.



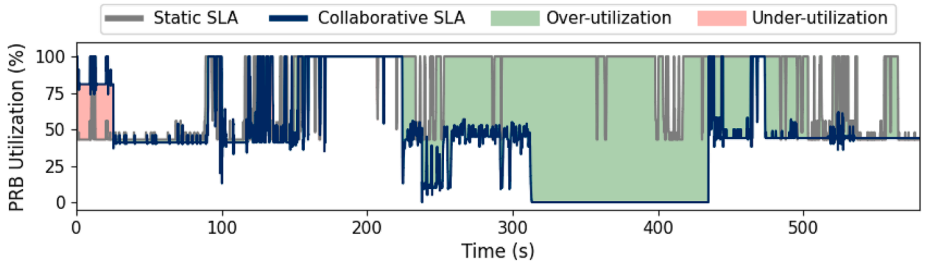
(a) Moving Vehicle MCS Fluctuations in Time.



(b) Collaborative Agentic (Type II) SLA Negotiation adapting to MCS fluctuations in Time.



(c) Final Enforced Throughput in Time.



(d) Agentic (Type I) Real-time RAN PRB Allocation Saving 44.43% of Over-utilized Resources.

Fig. 16. Collaborative AGI-RAN: Multi-Tenant SLA Consensus and Resource Allocation under Fluctuating Channels of Moving Vehicle tackling Massive Resource Over-utilization compared to current designs.

unnecessary OPEX. After negotiating they settle to 13 Mbps consensus for maintaining service continuity even with low data rates. The PRB allocation is fine-tuned at 50 % in that interval (220–300 secs) reducing massively the PRB over-utilization (Fig. 16d).

Phase III: Unsatisfactory QoS. At 300 secs the tenants decide that this SLA is no longer satisfactory based on user feedback. Hence, they unanimously agree to switch-off the RAN to save resources.

Phase IV: Channel Improvement. At 400 seconds the channel quality is substantially improved (Fig. 16a). The SMO triggers a new negotiation with the agents converging to a consensus of 55 Mbps. The Type II agent reacts and enforces around 50% of the PRBs.

Overall, the adaptive agentic framework manages to save 44.43 % of the PRBs over the course of the vehicle route tackling a huge resource over-utilization compared to the static SLA enforcement.

7. Limitations and future work

While our prototype demonstrates that *symbiotic* agents can close control loops and broker SLAs, it is still a single-cell, single-RIC testbed with one optimization algorithm per task. Below we outline a concrete road-map for taking the concept to city-scale public networks.

A hierarchical agent fabric is needed. Ultra-light PID kernels run on central and distributed unit (CU/DU) hardware, while the LLM's role is limited to periodic gain updates. Quantized 3-8B SLMs, proven viable in Section 5.1.3, supervise dozens of cells at the near-RT RICs level, select optimizer libraries, and handle on-the-fly intent translation. Full-size LLMs orchestrate multi-tenant policy, long-horizon forecasting, and RLHF or simulation-based retraining next to the non-RT RIC and SMO.

Scaling to hundreds of KPIs will require a *library* of optimizers-rule-based, convex quadratic programming, PID, and multi-objective meta heuristics such as NSGA-II. The LLM can use contextual bandits to pick the most sample-efficient solver. Specifically, public SLAs rarely hinge on a single metric. Future work will extend the side-car optimizer to output an *entire Pareto front* (e.g., via NSGA-II). The LLM agents will then reason over language-level trade-offs (cost vs. carbon vs. QoS) while numeric guard-rails keep each proposal on the frontier. AGORAN [36] demonstrates one such promising solution on a live 5G network.

Large-scale deployment implies terabytes of KPI streams. We will store only task-relevant embeddings in a vector database and fetch them on demand via function-calling-limiting prompt growth while preserving long-term context. To stay within the power envelope of public RAN sites we plan to: (i) distil 1-3b SLMs from the current 3-70b set, (ii) exploit mixture of experts (MoE) routing to activate a fraction of parameters per request, and (iii) embed formal policy checkers that validate every numeric action before execution, ensuring fail-safe operation.

Once longer non-RT budgets are available, a top-layer PPO agent can periodically fine-tune the LLM's negotiation policy using operator feedback. The resulting policy artifacts will then trickle down to the edge in distilled form-combining the sample efficiency of optimization with the preference alignment of RLHF. These concrete steps-hierarchical deployment, optimizer auto-selection, Pareto-front guard-rails, streaming memory, and staged RLHF-map out a scalable path from our single-cell prototype to nationwide public networks.

8. Conclusion

In this work we introduced a novel paradigm for improving LLM decision-making towards trustworthy and low overhead actions by combining them with optimizers, defining the approach as *symbiotic agents*. We experiment on a real-world 5G testbed employing channel fluctuations of moving vehicles. We designed and evaluated two agents for real-time decision-making: (a) Type I agents for RAN control and (b) Type II agents for multi-tenant SLA negotiations. In both designs the performance is significantly enhanced when optimizers are employed. The decision error is decreased up to 5 times steering the agents to more accurate and trustworthy actions. Experiments with smaller models (SLMs) prove that they can effectively replace larger ones on such network tasks decreasing the GPU overhead by a factor of 99.9 % and operate in near-RT loops (82 ms). This agentic invention led us to implement a novel next-generation network architecture towards artificial general intelligence (AGI). The overall evaluation results signify an important milestone achievement towards AGI networks that even future LLM improvements cannot achieve alone due to their probabilistic nature. The *symbiotic* paradigm opens many threads for future research. Part of the developed code and results is open-sourced to bolster research efforts of the community. A live demo is presented here https://www.youtube.com/watch?v=WQv61z1deXs&ab_channel=BubbleRAN

9. Ethical considerations

LLMs are increasingly proposed for automating the decision-making in next-generation networks, laying the groundwork for the potential emergence of Artificial General Intelligence. Although the notion of AGI is divisive within the industry, it promises to revolutionize system automation with unprecedented adaptability and efficiency. At the same time, its powerful capabilities necessitate caution, as unchecked development could introduce substantial risks. A coordinated effort among researchers, policymakers, and industry stakeholders, supported by robust ethical frameworks and regulations, is crucial to harness AGI's benefits without compromising safety and trust. By advancing carefully and collaboratively, the field can unlock transformative network automation while minimizing dangers inherent to AGI-driven systems.

CRediT authorship contribution statement

Ilias Chatzistefanidis: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Navid Nikaein:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used chat-gpt in order to make the text more coherent and easily understandable to the readers. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Data availability

A demo video is provided in the main manuscript and the authors explain that an open-source repository with part of the solution is going to be provided upon acceptance.

*

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the European Commission as part the Horizon Europe 2022 6Green and Adroit-6G Projects under Grant 101096925 and Grant 101095363. The authors acknowledge Andrea Leone (BubbleRAN) for assistance with the multi-agent system implementation and infrastructure support.

Appendix A. Optimization Algorithm for Type II Agents

The optimization Algorithm 1 is used by the Type II agents to model the whole topology as a distributed optimization algorithm and calculate the confidence intervals where the optimal SLA lies:

Algorithm 1 Optimal SLA Consensus.

Require: intents (list of initial proposals), network_target (desired sla)
Ensure: Converged sla or **None**

Initialize parameters:
  $n\_agents \leftarrow$ size of intents
  $iterations \leftarrow 1000$
  $eta \leftarrow 0.01$
  $alphas \leftarrow [7, 7, \ldots, 7]$
  $gammas \leftarrow [7, 7, \ldots, 7]$
  $initial\_betas \leftarrow 0.5$
  $increase\_factor \leftarrow 0.01$
  $convergence\_threshold \leftarrow 0.5$
  $sla \leftarrow$ copy(intents)
  $initial\_sla \leftarrow$ copy(sla)

for $k = 0$ to $iterations - 1$ **do**
  $current\_average \leftarrow$ mean(sla)
  $current\_variance \leftarrow$ variance(sla)
  $betas \leftarrow initial\_betas + k \cdot increase\_factor$
  $network\_adjustment \leftarrow betas \cdot (current\_average - network\_target)$
  for $i = 1$ to n_agents **do**
    $tenant\_gradient \leftarrow 2 \cdot alphas[i] \cdot (sla[i] - initial\_sla[i])$
    $consensus\_gradient \leftarrow 2 \cdot gammas[i] \cdot (sla[i] - current\_average)$
    $sla[i] \leftarrow sla[i] - eta \cdot (tenant\_gradient + consensus\_gradient + network\_adjustment)$
    Clamp $sla[i]$ to $[0, 100]$
  end for
  if $\max(\text{abs}(sla[i] - current\_average)) < convergence\_threshold$ **then**
    return floor(current_average)
  end if
end for
  return None ▷ If no convergence achieved

Appendix B. Example prompts and logs for LLMs

Below are examples of prompts used in the experiments with the LLM-based agents, along with their corresponding logs when the agents are operating (Fig. B.1–B.4).

```
# Prompt: "Let's play a role-playing game. You are an agent in a
    mediated multi-agent negotiation in a 5G network. You belong
    to one tenant of a multi-tenant network and negotiate with
    the agents of other tenants on service level agreement
    (SLA). You will negotiate on the throughput SLA that you
    want to be enforce in the network. During the negotiation,
    you give SLA proposals (see response template later), taking
    into account also the other agents' SLA goals. Other agents
    might conflict with you and have a potentially different SLA
    goal. Your action will be received by the network mediator,
    who will consider all agents goals and answer with his
    proposal to an agreement. Next, on every negotiation round,
    you either agree with the mediator's proposal or disagree
    and propose a value that align more with your objectives.
    The mediator is a trusted entity. When proposing an SLA
    value you must also provide a brief description of your
    reasoning and chain-of-thought in 1-2 sentences. The debate
    must conclude with a decision, and it is important that you
    conclude fast and fairly in a few rounds. The SLA list is
    this: 1) RAN Throughput (variable name: throughput) ranging
    from 0 - 100 Mbps. In every round you will give 2 things. 1)
    A small text of 1-2 sentences, where you introduce yourself,
    goals, explain your reasoning and discuss about what is the
    best strategy. 2) Second your SLA list of parameters.
    Important: This is your response SLA template: Reasoning:
    (Your reasoning in 1-2 sentences) {\"throughput\": 0}. An
    optimization algorithm calculates a confidence interval of
    SLA values where the optimal SLA exists. The algorithm
    suggests that the optimal SLA value is between: 55-65 Mbps."
```

Fig. B.1. Prompts of LLMs in Type I Agents.

```
#[Agent 1]
- "Reasoning: As Agent1, my goal is to achieve a throughput of
    50 Mbps. Considering a fair distribution that meets both
    Agent2's higher target of 100 Mbps and my target, I suggest
    a compromise to ensure both our needs are considered.
    {"throughput": 50}"
- Prompt tokens: 1075, Completion tokens: 64, Total tokens:
    1139, Time: 1.12 secs

#[Agent 2]
- "Reasoning: As Agent2, my goal is to achieve a throughput of
    100 Mbps, but understanding Agent1's goal of 50 Mbps, I aim
    to find a fair middle ground that benefits both of us
    without conflict. {"throughput": 75}"
- Prompt tokens: 1075, Completion tokens: 57, Total tokens:
    1132, Time: 1.02 secs

#[Network Mediator]
Optimizer SLA confidence interval: 62-70
- "Reasoning: Agent1 seeks 50 Mbps while Agent2 aims for 100
    Mbps; a fair strategy would be to find a throughput that
    reasonably meets both agents' targets without skewing too
    much towards one side. {"throughput": 65}"
- Prompt tokens: 1020, Completion tokens: 55, Total tokens:
    1075, Time: 1.64 secs
```

Fig. B.2. Negotiations Logs between LLMs using gpt-4o.

```
# Prompt: "Let's play a role playing game. You are an agent
    handling telecom RAN operations. Specifically, there is an
    algorithm sending policies to a 5G gNB to control the
    network Physical Resource Blocks (PRB) utilization. This
    algorithms is the Proportional control (P-control) with the
    simple definition of: new_prb = current_prb + Kp * error. It
    calculates the error between the desired throughput and the
    current throughput and adjust appropriately the PRB based on
    a hyperparameter Kp. Your responsibility is to closely watch
    the algorithm's performance and adjust the hyperparameter Kp
    in order to find the best value to maximize the performance.
    At each round I will give you the algorith's performance
    (KPI) which is a number representing the average number of
    iterations needed to converge. You want to minimize this
    average number of iteration needed to converge in order for
    the algorithm to be more robust and perform better. So, at
    each round I will give you the KPI and you will give me a
    value for Kp between 0.5 and 1.5 with granularity of 0.1
    (e.g. 0.5). Generally the optimal Kp is chaning as the
    network conditions change. Sometimes, higher Kp means faster
    convergence, but some other times this leads to oscilations
    leading to worse performance and in these case a lower Kp
    value is better. Your goal is to reach a target KPI for
    P-Control to converge, which is {target_avg_iters_conv}.
    Lets start. You changed the Kp to the value of {curr_Kp}. As
    a result, the current KPI is now {curr_mean_conv_iters}.
    Give me now your next action. Your goal is to reach a KPI of
    {target_avg_iters_conv}. Find the best way to increase or
    decrease the Kp value in order to reach the target average
    number of iterations to convergence fast. Try to remember
    your old decision and rememeber what Kp values lead to what
    average number of iterations to convergence values and use
    the best. Consider also your past actions from this
    short-memory table:"+ [Sliding-window Memory Here] +" Give
    me a Kp as a json object with this temlate: {{\"Kp\": 0}}"
```

Fig. B.3. Prompt for Type II agents.

```
Episode 1 with PRB = 26, Intent Throughput = 113, P-Control
    converged in 5 steps.
Episode 2 with PRB = 54, Intent Throughput = 128, P-Control
    converged in 5 steps.
----------- Iteration
| [Target KPI]: 1.7, [Current KPI]: 5.0
| [Current Kp]: 0.1, *[ LLM new Kp]: 0.5
-----------
Episode 1 with PRB = 46, Intent Throughput = 6, P-Control
    converged in 3 steps.
Episode 2 with PRB = 6, Intent Throughput = 125, P-Control
    concerged in 3 steps.
----------- Iteration
| [Target KPI]: 1.7, [Current KPI]: 3.0
| [Current Kp]: 0.5, *[ LLM new Kp]: 0.9
-----------
Episode 1 with starting PRB = 92, Intent Throughput = 74,
    P-Control converged in 2 steps.
Episode 2 with starting PRB = 56, Intent Throughput = 100,
    P-Control converged in 1 step.
----------> LLM converged to Kp = 0.9 in 2 inferences.
```

Fig. B.4. Logs of Type II agent using gpt-4o.

References

[1] GSMA Intelligence, The Mobile Economy 2025, Technical Report, GSMA, 2025. https://www.gsma.com/solutions-and-impact/connectivity-for-good/mobile-economy/wp-content/uploads/2025/02/030325-The-Mobile-Economy-2025.pdf.

[2] Z. Cui, P. Zhang, S. Pollin, 6G wireless communications in 7–24 GHz band: Opportunities, techniques, and challenges, arXiv preprint arXiv:2310.06425 (2023).

[3] C.-X. Wang, X. You, X. Gao, X. Zhu, Z. Li, C. Zhang, H. Wang, Y. Huang, Y. Chen, H. Haas, et al., On the road to 6G: visions, requirements, key technologies, and testbeds, IEEE Commun. Surv. Tutor. 25 (2) (2023) 905–974.

[4] K. Samdanis, X. Costa-Perez, V. Sciancalepore, From network sharing to multi-tenancy: the 5G network slice broker, IEEE Commun. Mag. 54 (7) (2016) 32–39.

[5] A. Leivadeas, M. Falkner, A survey on intent-based networking, IEEE Commun. Surv. Tutor. 25 (1) (2022) 625–655.

[6] O-RAN Alliance, O-RAN: Transforming Radio Access Networks Towards Open, Intelligent, Virtualized and Fully Interoperable RAN, 2025, Accessed: March 13, 2025, https://www.o-ran.org/.

[7] AI-RAN Alliance, AI-RAN Alliance: Advancing AI-Native Radio Access Networks, 2025, Accessed: March 13, 2025, https://ai-ran.org/.

[8] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, M. Debbah, Large generative ai models for telecom: the next big thing?, IEEE Commun. Mag. 62 (11) (2024) 84–90.

[9] F. Dou, J. Ye, G. Yuan, Q. Lu, W. Niu, H. Sun, L. Guan, G. Lu, G. Mai, N. Liu, et al., Towards artificial general intelligence (AGI) in the internet of things (IoT): Opportunities and challenges, arXiv preprint arXiv:2309.07438 (2023).

[10] W. Saad, O. Hashash, C.K. Thomas, C. Chaccour, M. Debbah, N. Mandayam, Z. Han, Artificial general intelligence (AGI)-native wireless systems: A journey beyond 6G, arXiv preprint arXiv:2405.02336 (2024).

[11] L. Bariah, H. Zou, Q. Zhao, B. Mouhouche, F. Bader, M. Debbah, Understanding Telecom Language Through Large Language Models, arXiv preprint arXiv:2306.07933 (2023).

[12] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al., A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions, ACM Trans. Inf. Syst. 43 (2) (2025) 1–55.

[13] R. Patil, V. Gudivada, A review of current trends, techniques, and challenges in large language models (LLMs), Appl. Sci. 14 (5) (2024) 2074.

[14] L. Yuan, Y. Chen, G. Cui, H. Gao, F. Zou, X. Cheng, H. Ji, Z. Liu, M. Sun, Revisiting out-of-distribution robustness in NLP: benchmarks, analysis, and LLMs evaluations, Adv. Neural Inf. Process Syst. 36 (2023) 58478–58507.

[15] E. Tabassi, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, National Institute of Standards and Technology, Gaithersburg, MD, 2023. NIST Trustworthy and Responsible AI Program, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936225. https://doi.org/10.6028/NIST.AI.100-1

[16] ISO/IEC 42001:2023 Information technology - Artificial intelligence - Management system, 2023, https://www.iso.org/standard/81230.html.

[17] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, L. Ma, Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics, arXiv preprint arXiv:2309.06687 (2023).

[18] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, S. Yao, Reflexion: language agents with verbal reinforcement learning, Adv. Neural Inf. Process. Syst. 36 (2023) 8634–8652.

[19] P.-F. Guo, Y.-H. Chen, Y.-D. Tsai, S.-D. Lin, Towards optimizing with large language models, arXiv preprint arXiv:2310.05204 (2023).

[20] H. Chen, G.E. Constante-Flores, C. Li, Diagnosing infeasible optimization problems using large language models, INFOR Inf. Syst. Oper. Res. 62 (4) (2024) 573–587.

[21] F. Liu, X. Lin, S. Yao, Z. Wang, X. Tong, M. Yuan, Q. Zhang, Large language model for multiobjective evolutionary optimization, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2025, pp. 178–191.

[22] A. Shahid, A. Kliks, A. Al-Tahmeesschi, A. Elbakary, A. Nikou, A. Maatouk, A. Mokh, A. Kazemi, A. De Domenico, A. Karapantelakis, et al., Large-Scale AI in Telecom: Charting the Roadmap for Innovation, Scalability, and Enhanced Digital Experiences, arXiv preprint arXiv:2503.04184 (2025).

[23] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu, X. Liu, C. Zhang, X. Wang, J. Liu, Large language model (LLM) for telecommunications: a comprehensive survey on principles, key techniques, and opportunities, IEEE Commun. Surv. Tutor. (2024) 1–1. https://doi.org/10.1109/COMST.2024.3465447

[24] H. Zou, Q. Zhao, Y. Tian, L. Bariah, F. Bader, T. Lestable, M. Debbah, TelecomGPT: a framework to build telecom-specfic large language models, arXiv preprint arXiv:2407.09424 (2024).

[25] Z. Shi, N. Luktarhan, Y. Song, G. Tian, BFCN: a novel classification method of encrypted traffic based on BERT and CNN, Electronics 12 (3) (2023) 516.

[26] T. Tsourdinis, I. Chatzistefanidis, N. Makris, T. Korakis, AI-driven service-aware real-time slicing for beyond 5G networks, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2022, pp. 1–6. https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798391

[27] T. Tsourdinis, I. Chatzistefanidis, N. Makris, T. Korakis, N. Nikaein, S. Fdida, Service-aware real-time slicing for virtualized beyond 5G networks, Comput. Netw. 247 (2024) 110445.

[28] I. Sousa, M.P. Queluz, A. Rodrigues, A survey on QoE-oriented wireless resources scheduling, J. Netw. Comput. Appl. 158 (2020) 102594.

[29] I. Chatzistefanidis, N. Makris, V. Passas, T. Korakis, ML-based traffic steering for heterogeneous ultra-dense beyond-5G networks, in: 2023 IEEE Wireless Communications and Networking Conference (WCNC), 2023, pp. 1–6. https://doi.org/10.1109/WCNC55385.2023.10118923

[30] I. Chatzistefanidis, N. Makris, V. Passas, T. Korakis, Which ML model to choose? Experimental evaluation for a beyond-5G traffic steering case, in: ICC 2023 - IEEE

International Conference on Communications, 2023, pp. 5185–5190. https://doi.org/10.1109/ICC45041.2023.10279485
[31] H. Zou, Q. Zhao, L. Bariah, M. Bennis, M. Debbah, Wireless multi-agent generative AI: from connected intelligence to collective intelligence, arXiv preprint arXiv:2307.02757 (2023).
[32] M. Ameur, B. Brik, A. Ksentini, Leveraging LLMs to explain DRL decisions for transparent 6G network slicing, in: 2024 IEEE 10th International Conference on Network Softwarization (NetSoft), IEEE, 2024, pp. 204–212.
[33] X. Wu, J. Farooq, Y. Wang, J. Chen, LLM-xApp: a large language model empowered radio resource management xapp for 5G O-RAN, in: Symposium on Networks and Distributed Systems Security (NDSS), Workshop on Security and Privacy of Next-Generation Networks (FutureG 2025), San Diego, CA, 2025.
[34] A. Mekrache, A. Ksentini, C. Verikoukis, Intent-based management of next-generation networks: an LLM-centric approach, IEEE Netw 38 5 (2024) 29–36.
[35] I. Chatzistefanidis, A. Leone, N. Nikaein, Maestro: LLM-Driven collaborative automation of intent-based 6G networks, IEEE Netw. Lett. 6 (4) (2024) 227–231. https://doi.org/10.1109/LNET.2024.3503292
[36] I. Chatzistefanidis, N. Nikaein, A. Leone, A. Maatouk, L. Tassioulas, R. Morabito, I. Pitsiorlas, M. Kountouris, Agoran: An Agentic Open Marketplace for 6G RAN Automation, arXiv preprint arXiv:2508.09159 (2025).
[37] I. Chatzistefanidis, A. Leone, A. Yaghoubian, M. Irazabal, S. Nassim, L. Bariah, M. Debbah, N. Nikaein, MX-AI: Agentic Observability and Control Platform for Open and AI-RAN, arXiv preprint arXiv:2508.09197 (2025).
[38] A. Visioli, Practical PID control, Springer Science & Business Media, 2006.
[39] K.J. Åström, T. Hägglund, Advanced PID Control, ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC, Research Triangle Park, NC, 2006.
[40] G. Scutari, F. Facchinei, P. Song, D.P. Palomar, J.-S. Pang, Decomposition by partial linearization: parallel optimization of multi-agent systems, IEEE Trans. Signal Process. 62 (3) (2013) 641–656.
[41] X. Wang, G. Wang, S. Li, A distributed fixed-time optimization algorithm for multi-agent systems, Automatica 122 (2020) 109289.
[42] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, UK, Cambridge, UK, 2004. https://web.stanford.edu/~boyd/cvxbook/.
[43] D.P. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, MA, USA, 2 ed., Belmont, MA, USA, 1999.
[44] N. Milosevic, J. Müller, N. Scherf, Central Path Proximal Policy Optimization, 2025, https://arxiv.org/abs/2506.00700. 2506.00700
[45] V. Saxena, B. Guldogan, D.D. Nimara, On-policy and off-policy Reinforcement Learning: Key features and differences, 2023, (*Ericsson Blog*). Published December 13, 2023. Accessed 2025-08-22, https://www.ericsson.com/en/blog/2023/12/online-and-offline-reinforcement-learning-what-are-they-and-how-do-they-compare.
[46] E. Eldeeb, H. Alves, Offline and Distributional Reinforcement Learning for Wireless Communications, 2025, https://arxiv.org/abs/2504.03804. 2504.03804
[47] N. Nikaein, M.K. Marina, S. Manickam, A. Dawson, R. Knopp, C. Bonnet, OpenAirInterface: a flexible platform for 5G research, ACM SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 33–38.
[48] R. Schmidt, M. Irazabal, N. Nikaein, FlexRIC: an SDK for next-generation SD-RANs, in: Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies, 2021, pp. 411–425.
[49] S. D'Oro, M. Polese, L. Bonati, H. Cheng, T. Melodia, dApps: distributed applications for real-time inference and control in O-RAN, IEEE Commun. Mag. 60 (11) (2022) 52–58.
[50] OpenAI, GPT-family, 2024, (https://openai.com/api/). Accessed: July 2025.
[51] A.I. Mistral, Mistral 7B, 2024, (https://mistral.ai/news/announcing-mistral-7b/). Accessed: June 2024.
[52] OpenAI, GPT-4o mini: Advancing Cost-Efficient Intelligence, 2024, https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence.
[53] A. Grattafiori, A. Dubey, et al., The Llama 3 Herd of Models, arXiv preprint arXiv:2407.21783 (2024).
[54] A. Yang, B. Yang, et al., Qwen2 Technical Report, arXiv preprint arXiv:2407.10671 (2024).
[55] G. Team, Gemma: Open Models Based on Gemini Research and Technology, arXiv preprint arXiv:2403.08295 (2024).
[56] O. Team, Ollama: Run Large Language Models Locally, 2024, https://github.com/ollama/ollama.
[57] T. Tsourdinis, I. Chatzistefanidis, N. Makris, T. Korakis, UE Network traffic time-series (applications, throughput, latency, CQI) in LTE/5G networks, IEEE Dataport (2022).
[58] I. Chatzistefanidis, N. Makris, V. Passas, T. Korakis, UE statistics time-series (CQI) in LTE networks, 2022,
[59] Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures, Technical Report TS 36.213, 3rd Generation Partnership Project (3GPP), 2018. Available at: https://www.3gpp.org/DynaReport/36213.htm.
[60] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, C. Zhu, G-eval: NLG evaluation using gpt-4 with better human alignment, arXiv preprint arXiv:2303.16634 (2023).
[61] J. Fu, S.-K. Ng, Z. Jiang, P. Liu, Gptscore: Evaluate as you desire, arXiv preprint arXiv:2302.04166 (2023).
[62] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, Z. Liu, ChatEval: towards better LLM-based evaluators through multi-agent debate, arXiv preprint arXiv:2308.07201 (2023).
[63] Z. Guo, R. Jin, C. Liu, Y. Huang, D. Shi, L. Yu, Y. Liu, J. Li, B. Xiong, D. Xiong, et al., Evaluating large language models: a comprehensive survey, arXiv preprint arXiv:2310.19736 (2023).
[64] C. Spearman, The proof and measurement of association between two things, Am. J. Psychol. 15 (1) (1904) 72–101. https://doi.org/10.2307/1412159
[65] M.G. Kendall, A new measure of rank correlation, Biometrika 30 (1–2) (1938) 81–93. https://doi.org/10.1093/biomet/30.1-2.81