Holistic and Adaptive Agentic AI Orchestration for Secure and Efficient IoT Environment Management

Graziano Rizzo
DIMES, University of Calabria
Via P. Bucci, 87036 Rende (CS), Italy
graziano.rizzo@dimes.unical.it

Mattia Giovanni Spina
DIMES, University of Calabria
Via P. Bucci, 87036 Rende (CS), Italy
CNIT – National Inter-University Consortium for Telecommunications, Parma, Italy
mattiagiovanni.spina@dimes.unical.it

Mauro Tropea
DIMES, University of Calabria
Via P. Bucci, 87036 Rende (CS), Italy
CNIT – National Inter-University Consortium for Telecommunications, Parma, Italy
m.tropea@dimes.unical.it

Azzedine Boukerche
PARADISE Research Laboratory
EECS, University of Ottawa
Ottawa, Canada
aboukerc@uOttawa.ca

Floriano De Rango
DIMES, University of Calabria
Via P. Bucci, 87036 Rende (CS), Italy
CNIT – National Inter-University Consortium for Telecommunications, Parma, Italy
derango@dimes.unical.it

Abstract*—Internet of Things (IoT) systems increasingly comprise heterogeneous, interconnected, resource-constrained devices with diverse security and service requirements, making security and operational efficiency joint design objectives. Existing solutions typically address these concepts as isolated dimensions of the problem, rarely providing a unified formal model for coordinated decision-making under evolving operating and network conditions. To address this limitation, this paper leverages Agentic AI to propose an orchestration framework in which multiple agents jointly act to collaboratively manage security and operational conditions of heterogeneous IoT environments, seamlessly accounting for competing and complementary requirements. The proposed Agentic AI-based framework builds a multi-view IoT state descriptor from distributed observations, decomposes the control logic into role-specific agents with explicit beliefs, goals, action spaces, and outputs, and formulates orchestration as a constrained decision problem in which security and policy requirements are enforced as hard admissibility constraints. The contribution of this paper, therefore, aims at translating the Agentic AI paradigm into an implemented comprehensive IoT orchestration framework capable of jointly adapting security and efficiency under evolving and heterogeneous system conditions. The proposal, validated under a Software-Defined Network (SDN)-enabled IoT scenario, demonstrates its benefits and adaptability against static, security-only, and service-only orchestration strategies stressed under different network and attack conditions. Particularly, considering the best-performing competing strategy for each KPI, the proposal reduces average delay by 91.6-92.5%, packet loss by 81.1-90.3%, and security violations by 34.7-59.7% across the attack scenarios, while maintaining a control overhead comparable to that of the security-only approach.

***Index Terms*—Agentic AI, Internet of Things, Multi-Agent Coordination, IoT Security, Network Security, Programmable Networks**

I. Introduction

The Internet of Things (IoT) is evolving from collections of isolated sensing devices toward programmable service systems in which devices, gateways, edge nodes, cloud services, and control-plane functions cooperate to support distributed and complex applications [1]. These environments are characterized by heterogeneous computational capabilities and application-level requirements that may change over time [2]. In this context, management decisions cannot be reduced to a single control dimension. For instance, a routing update may reduce delay while increasing security exposure, whereas a mitigation action may isolate a suspicious node at the cost of service availability [3], [4]. IoT orchestration should therefore be treated as a joint security-and-efficiency control problem where security should not be considered an external add-on activated only after attack detection, but a design constraint embedded into every decision. Similarly, efficiency should preserve service-level objectives while respecting energy, computational, communication, and policy constraints [5]. This observation motivates the need for an adaptive framework capable of observing heterogeneous system conditions, abstracting them into a shared operational state, reasoning over competing objectives, and enforcing safe actions while continuously learning from runtime feedback. Recently, the agentic AI paradigm has emerged as a promising solution for autonomous network orchestration and control [6]. Unlike monolithic AI models, agentic architectures decompose the decision process into specialized entities – namely AI agents – that collaboratively implement the *perceive* $\rightarrow$ *reason* $\rightarrow$ *act* $\rightarrow$ *learn* $\rightarrow$ *collaborate* closed-loop [7] to pursue a common optimization objective. Such a decomposition is particularly suitable for IoT systems, where security assessment, quality of service, resource allocation, and policy enforcement rely on complementary information and may generate conflicting recommendations [8]. However, existing agentic approaches generally address these dimensions independently, without

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

providing a unified decision process. To address this limitation, this paper proposes an Agentic AI framework for adaptive and holistic IoT orchestration. The framework is designed as a cooperative multi-agent architecture composed of role-specific agents for security-policy evaluation, service assurance, resource and function placement, and global orchestration. The proposal builds a unified Multi-View State Descriptor (MSD) combining device, network, flow, security, and contextual information. Role-specific agents evaluate security policies, service conditions, and resource-placement feasibility, while an Orchestration Agent aggregates their outputs and selects the final action plan, ensuring seamless and comprehensive network and security management. The proposed solution is implemented and evaluated in a containerized Software-Defined Network (SDN)-enabled IoT environment. A comprehensive experimental campaign, under variable network traffic profiles and different types of attacks and intensities, shows that jointly orchestrating multiple dimensions consistently improves both security and system operational conditions, when compared against a *static*, *security-only*, and *service-only* orchestration. The remainder of this paper is organized as follows: Section II provides an overview of the related works. Section III delves into the details of the proposed Agentic AI orchestration framework. Section IV and Section V describe the experimental setup and the results, respectively. Finally, Section VI concludes the manuscript.

II. Related Works

6G is expected to strongly improve network orchestration and management, providing the network with self-adaptive capabilities for both security and resource utilization requirements [9]–[11]. In this perspective, the integration of AI, and specifically Agentic AI, into the network constitutes an enabling paradigm to achieve such a transformative leap, especially in the evolving and complex IoT context [7], in which static security configuration fails in adapting to changing attack conditions and strategies [12].

In [13], the authors proposed to integrate zero-touch and zero-trust principles along with blockchain and agentic AI for a secure 6G Network. However, the work remains at an early stage and primarily focuses on the security dimension, without jointly addressing service efficiency and resource orchestration. Moreover, the integration of blockchain introduces additional overhead that may limit its applicability in resource-constrained and highly dynamic environments. Agentic AI and GenAI are used in [14] to address effective energy management in IoT networks, enabling autonomous adaptation to real-time environmental and network conditions. The concept of Industry 5.0 appears in [15], in which the authors envisioned AI-enhanced future factories in which cyber-physical systems and humans can seamlessly collaborate by means of an Agentic AI-driven management. Specifically, it relies on Large Language Models (LLMs) to optimally and dynamically orchestrate technology pipelines that best suit the factory environment. Nevertheless, the proposal mainly addresses the operational orchestration of industrial and technology pipelines, neglecting the complexities underlying the heterogeneity of the network infrastructure and the variability of communication, resource, and security constraints. A distributed agentic intelligence layer for future 6G networks is proposed in [16]. The main objective of this work is the autonomous and adaptive management of cross-domain services and networks. AgentRAN is proposed in [17]. It exploits agentic AI to interpret natural-language intents and negotiate strategies for autonomous control of open 6G networks.

Main Contributions

None of the reviewed works jointly exploits Agentic AI to address multiple, potentially competing dimensions, including security, resource utilization, and network performance. Although Agentic AI is increasingly adopted in the reviewed solutions to enhance autonomous management, these solutions typically focus on a single operational dimension, not considering a comprehensive framework that can relate and exploit information from multiple operational dimensions to provide a seamless and effective orchestration. To fill these gaps, this manuscript proposes a unified framework in which multiple and role-specific agents collaborate and share their observations to jointly orchestrate security, network requirements, and resource management, enabling a holistic and adaptive orchestration tailored to heterogeneous and dynamic IoT environments. In this direction, a cooperative multi-agent architecture is designed and implemented in which role-specific agents share complementary system views and coordinate their recommendations through a closed-loop decision process. Finally, the framework is implemented and evaluated in an SDN-enabled IoT environment, showing better performance than *static*, *security-only*, and *network-only* strategies, demonstrating the benefits of jointly combining observations from multiple dimensions to account for an effective and comprehensive orchestration.

III. Agentic AI-based Holistic Orchestration Framework for IoT

The IoT environment is modeled as a time-varying system in which the set of nodes (i.e., IoT devices, edge nodes, cloud components, and remote control functions, etc.) available in the system at time t is denoted as $D(t)$. Following this formulation, each node $d \in D(t)$ is characterized by a resource vector $\mathbf{r}_d(t)$ represented as:

$$\mathbf{r}_d(t) = [cpu_d(t), mem_d(t), bw_d(t), en_d(t), sto_d(t)]\,, \quad (1)$$

where cpu_d, mem_d, bw_d, en_d, and sto_d represent available processing, memory, bandwidth, residual energy, and storage resources, respectively. The interaction among IoT entities is modeled through a set of logical or physical communication links available at time t, denoted as $L(t)$. Each link $c_{ij}(t) \in L(t)$ connects two nodes $i, j \in D(t)$ and is associated with a vector $\mathbf{l}ij(t)$ which represents the operational status of the link:

$$\mathbf{l}_{ij}(t) = [q_{ij}(t), \sigma_{ij}(t)]\,, \quad (2)$$

where $q_{ij}(t)$ denotes the estimated link quality, representing all the observed performance metrics such as the available bandwidth, the observed delay, the signal quality, etc., while $\sigma_{ij}(t)$ represents the communication metrics such as the traffic characterization and the amount and details of data exchanged between nodes. The set of nodes and communication links does not directly imply that the service flows are fully known a priori, since in realistic IoT environments, service flows are often only partially observable and must be inferred from the traffic observed between nodes on the different logical communication links. For this reason, the first entry point of the intelligent orchestration layer is a flow-based feature extraction stage that analyzes the traffic observations and reconstructs a compact view of the active or expected service demands. Let $\mathcal{O}_{tr}(t)$ denote the set of traffic-related observations collected at time t over the nodes in $D(t)$ and the links in $L(t)$. These observations included in $\mathcal{O}_{tr}(t)$ may include packet counters, source and destination identifiers, protocol information, inter-arrival times, queuing indicators, forwarding metadata, application identifiers (when available), and control-plane information. Based on these observations, the feature extractor derives an estimated set of service flows, namely $\mathcal{F}(t)$, in which each inferred flow $f \in \mathcal{F}(t)$ represents an active or emerging communication demand between IoT entities, applications, or control functions. In this context, it is worth noticing that the objective is not to model the IoT system as a static collection of known flows: in fact, nodes and links provide the observable substrate over which traffic is monitored, while the flow-aware feature extraction (FE) stage produces a dynamic representation of service demands that becomes one of the inputs used to build the global state artifact consumed by the *Intelligent Layer*. This artifact is called *Multi-View State Descriptor (MSD)*, and it summarizes the current operating condition of the IoT system; thus, it is defined as follows:

$$\mathit{MSD}(t) = \{\mathcal{X}dev(t), \mathcal{X}net(t), \mathcal{X}flow(t), \mathcal{X}sec(t), \mathcal{X}ctx(t)\} \tag{3}$$

where $\mathcal{X}dev(t)$ is the device view that represents the condition of the devices and edge resources; $\mathcal{X}net(t)$ is the network view that represents the perceived state of links, paths and connectivity; $\mathcal{X}flow(t)$ is the flow view that describes the active or inferred service demands and their delivery conditions; $\mathcal{X}sec(t)$ is the security view, where the current security posture is specified, including anomalies, exposure and trust-related indicators; Finally, $\mathcal{X}ctx(t)$ represents the context view, that specifies external or policy-related information that describes how the other views should be interpreted one to each other. The idea behind this approach is that each view on which the MSD is based captures a different dimension of the IoT system that is relevant for the orchestration of the system itself, acting therefore as the "interface" between the feature extraction stage and the *Intelligent Layer*. For this reason, the MSD is maintained over time to reflect the dynamicity and heterogeneity of IoT ecosystems, in which the effect of a previous orchestration decision may not reflect current requirements. The update performed on the MSD at time t follows, therefore the rule:

$$\mathit{MSD}(t) = \Phi\left(\mathit{MSD}(t-1), \mathcal{O}_{tr}(t), \mathcal{F}(t), FB(t-1)\right), \tag{4}$$

where $\Phi(\cdot)$ is the aggregation function, $\mathcal{O}(t)$ is the set of observations collected at time t, $\mathcal{F}(t)$ is the set of active or inferred service flows produced by the flow-aware feature extraction stage and $FB(t-1)$ is the feedback generated after the previous orchestration step. All these components allow the MSD to incorporate the observed effects of previous decisions, capturing the temporal evolution of the system and the operational consequences of past actions. It represents the final decision-oriented artifact consumed by the *Intelligent Layer*.

A. Intelligent Layer

Each agent A_i that composes the *Intelligent Layer* is modeled as $A_i = \langle O_i, B_i, G_i, S_i, \Pi_i, Y_i, \rho_i \rangle$, where: O_i is the observation space; B_i is the internal belief state; G_i is the set of goals; S_i is the action space; Π_i is the decision policy; Y_i is the output artifact generated by the agent; and ρ_i is the feedback or utility signal used by the agent itself to revise future decisions. The idea behind this definition is to guarantee that each agent has explicit inputs, outputs, goals, and personal decision logic, making the *Intelligent Layer* a set of role-specific agents that operate on specific contexts and information with role-specific objectives. For this purpose, a set of four agents that cooperate on specific information exchanged horizontally in the same layer is used in this framework.

The ***Security Policies Evaluation Agent*** **(SPE-A)** evaluates the current security posture of the IoT system and verifies whether the observed or inferred flows are compatible with the security requirements of the system. Its main observation space is:

$$O_{SPE}(t) = \{\mathcal{X}sec(t), \mathcal{X}flow(t), \mathcal{X}net(t)\}, \tag{5}$$

with a primary focus on the security and flow views. The network view is included because some security violations such as suspicious paths, abnormal forwarding patterns, or routing manipulation are visible only by correlating security signals with connectivity information. The goals of SPE-A are to compute a risk estimation on the actual state of the system, to evaluate the access policies that are currently running in the system, to perform the trust assessment and consequently provide the mitigation recommendation (e.g. by isolating suspicious entities or flows). The output of the SPE-A is the *Security Evaluation Report (SER)* that includes the set of monitored assets (associated with their risk indicator), the set of plausible threats, and the set of candidate security mechanisms to apply.

The ***Service Policies Assurance Agent*** **(SPA-A)** evaluates whether the current and inferred service flows can satisfy the service level requirements associated with delay, reliability, and priority; therefore, its observation space is:

$$O_{SPA}(t) = \{\mathcal{X}flow(t), \mathcal{X}net(t), \mathcal{X}ctx(t)\}, \tag{6}$$

since its primary focus must be on the flow and network views. This agent aims to preserve service continuity and delivery reliability while controlling delay, allowing a controlled degradation of non-critical services when full satisfaction is infeasible. It produces a *Service Report* (SR) summarizing flow-level service stress, requirements, and the set of candidate service-level adaptations.

The ***Resources and Placement Agent*** **(RP-A)** evaluates whether the actions or functions required by a candidate orchestration decision can be supported by the available infrastructure. For this reason, its observation space is defined as:

$$O_{RP}(t) = \{\mathcal{X}dev(t), \mathcal{X}flow(t), \mathcal{X}net(t)\}, \tag{7}$$

The network view is considered essential in this case because placement decisions are affected by connectivity, path quality and overall communication cost. The RP-A aims to assess resource feasibility, function placement, estimate resource allocation and verify the actual deployability of the proposed action. The RP-A generates an output named *Resource and Placement Report (RPR)* that includes the operational conditions of the nodes (e.g., current load and energy condition), plus the feasible placement set for functions or actions.

The ***Orchestration Agent*** **(OR-A)** is the central decision entity of the *Intelligent Layer* that operates to synthesize a final orchestration decision that is consistent with the operational context, compliant with policies, feasible with respect to resources, and effective with respect to security and service objectives. For this reason, the OR-A receives both the MSD (from the Control Layer) and the reports generated by the three aforementioned agents; thus, its observational space can be formalized as:

$$O_{OR}(t) = \{\textit{MSD}(t), SER(t), SR(t), RPR(t)\}. \tag{8}$$

Due to the importance of the OR-A in the proposed framework, since this agent must simultaneously process the heterogeneous reports produced by the first-level agents, reconcile their possibly competing objectives, preserve a global view of the system through $\textit{MSD}(t)$ and guarantee that any candidate decision complies with the contextual and policy constraints encoded in $\mathcal{X}_{ctx}(t)$, this agent is modeled as a composite agent with the idea to internally decompose the Orchestration Agent into specialized reasoning, validation, optimization and feedback components. This formal structure of this agent can be represented as:

$$A_{OR} = \left\langle A_{OR}^{LLM}, A_{OR}^{VAL}, A_{OR}^{OPT}, A_{OR}^{FB} \right\rangle, \tag{9}$$

where A_{OR}^{LLM} is the semantic reasoning component, A_{OR}^{VAL} is the policy and constraint validation component, A_{OR}^{OPT} is the decision selection component and A_{OR}^{FB} is the feedback-management component. It is worth noticing that each of these sub-components has its specific role and responsibilities: the semantic component A_{OR}^{LLM} is responsible for interpreting the context view $\mathcal{X}ctx(t)$ and checking whether the recommendations provided by the other agents are coherent with the constraints imposed by the context, domain-specific rules and administrative policies; The validation component A_{OR}^{VAL} transforms this interpretation into explicit admissibility checks; The optimization component A_{OR}^{OPT} ranks the admissible candidate plans according to service efficiency, resource usage and expected operational benefits; Finally, A_{OR}^{FB} manages the bidirectional interaction with the underlying agents by requesting revisions when their outputs are incomplete, conflicting or not aligned with the context.

The OR-A generates a set of candidate plans, namely $\mathcal{P}(t)$ and each candidate plan $p \in \mathcal{P}(t)$ is evaluated against a context-aware admissibility predicate derived from $\mathcal{X}_{ctx}(t)$ which includes the set of security constraints derived from $SER(t)$, the set of service constraints derived from $SAR(t)$ and the set of resource and placement constraints derived from $RPR(t)$. The plans that result admissible among the candidate ones are grouped in a set of admissible plans, namely $P_{adm}(t)$, from which the final plan is selected by the OR-A as:

$$p^{\star}(t) = \arg\min *p \in \mathcal{P}_{adm}(t) J(p,t), \tag{10}$$

where $J(p,t)$ is a cost function that ranks only plans that already satisfy security, service, resource, and context constraints. Therefore, optimization is performed only inside the admissible region and does not trade policy or safety compliance against performance. If $\mathcal{P} * adm(t) = \emptyset$, the Orchestration Agent does not enforce any unsafe decision and, instead, it activates a revision step by sending a set of feedback requests, namely $Req(t)$, to one or more of the underlying agents (based on which aspect of the plan should be optimized). In fact, these requests are sent to ask the agents to revise their recommendations, relax non-critical objectives when allowed by policy, provide additional confidence information, or generate alternative proposals. The final output of the Orchestration Agent is

$$Y_{OR}(t) = \{p^{\star}(t), E(t), Req(t), FB(t)\}, \tag{11}$$

where $p^{\star}(t)$ is the selected plan, $E(t)$ is the enforcement command sent to the control plane, $Req(t)$ is the optional feedback sent to the underlying agents and $FB(t)$ is the feedback information used to update the MSD and future agent decisions (that represents $FB(t-1)$ in Eq.4).

B. Interfaces and AI Models

The proposed *Intelligent Layer* requires structured interfaces among the first-level agents, the Orchestration Agent, and the control plane.

The first interface that is found in the intelligent layer is the *State-to-Agent interface (S2A)* that delivers the *MSD(t)* from the control plane to all the first-level agents:

$$I_{CP\to A_i}^{S2A} : \textit{MSD}(t) \to O_i(t), \quad A_i \in \{A_{SPE}, A_{SPA}, A_{RP}\}. \tag{12}$$

where $O_i(t)$ is the observation space of the i-th agent, as described in subsection III-A. This interface extracts from the MSD the views that are relevant to each first-level agent: The Security Policies Evaluation Agent mainly receives security, flow, and network information; the Service Policies Assurance

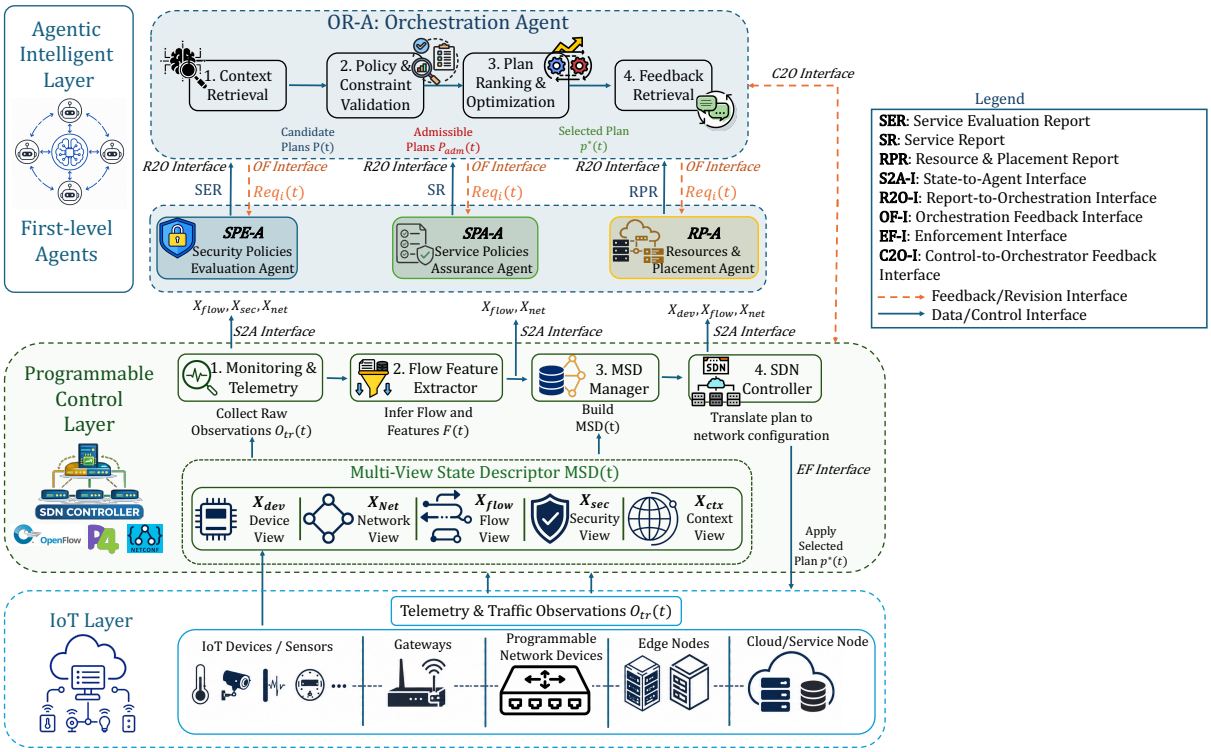


Fig. 1: Architecture of the proposed framework with the agents and the artifacts involved in the execution loop.

Agent mainly receives flow, network, and context information; the Resource and Placement Agent mainly receives device, flow, and network information. For these reasons, the interface may be implemented as a structured query over the MSD(t) artifact or a set of APIs exposing the selected views for each agent based on its needs.

The second interface is the *Report-to-Orchestrator (R2O)* interface between first-level agents and the Orchestration Agent, formalized as:

$$I^{R2O}_{A_i \to OR} : Y_i(t) \to A_{OR}, \quad A_i \in \{A_{SPE}, A_{SPA}, A_{RP}\}. \tag{13}$$

This interface carries structured reports (i.e., the *SER*, the *SR*, and the *RPR*, represented as $Y_i(t)$) generated by the first-level agents to the Orchestration Agent. Each report should include at least the assessed entities, the detected condition, the recommended action or constraint, the expected benefit, the confidence level, and the possible side effects in a format that is machine-readable and allows the Orchestration Agent to compare recommendations generated by the different agents. The third interface is the so-called *Orchestration Feedback (OF)* interface:

$$I^{OF}_{OR \to A_i} : Req_i(t) \to A_i, \quad A_i \in \{A_{SPE}, A_{SPA}, A_{RP}\}. \tag{14}$$

It allows the OR-A to request revisions from a first-level agent when a report is incomplete, conflicts with another report, violates contextual constraints, or does not provide enough confidence for enforcement, as already mentioned in subsection III-A. On this interface, for example, the Orchestration Agent may ask the SPA agent to propose a less aggressive prioritization policy if the context view indicates that mission-critical traffic must not be delayed, or it may ask the RP agent to compute an alternative placement if the proposed node is not trusted enough for a security-sensitive function.

The fourth interface is the *Enforcement (EF)* interface toward the IoT Layer:

$$I^{EF}_{CP \to IoT} : p^\star(t) \to E(t), \tag{15}$$

where $E(t)$ is the configuration that must be applied to the network (e.g., routing updates, flow-table rules, rate-limiting, access-control policies, etc...).

Finally, the *Control-to-Orchestrator (C2O)* feedback interface connects the control plane to the Orchestration agent in a bidirectional way.

$$I^{C2O}_{OR \leftrightarrow CP} : FB(t) \leftrightarrow MSD(t+1). \tag{16}$$

This interface carries information about the effects of the enforced plan. For instance, it can show evidence on whether or not a mitigation was successfully installed, whether congestion was reduced, whether SLA violations were solved, or whether malicious flows were blocked. The control later, then, uses this interface to send MSD artifacts to the OR-A, to provide the orchestration entity with all the information that is contained in the MSD (i.e., it gives the OR-A complete knowledge of the formalization artifact that describes the network).

IV. Experimental Setup & Agentic Layer Design Description

This section delves into the details of the experimental setup and agents used to develop the orchestration framework.

A. Experimental Setup

The experimental environment is implemented as a containerized IoT testbed in which IoT devices and agentic modules are deployed as independent Docker containers while link delay, packet loss, bandwidth limitation, and transient disconnections are injected through the console command *tc/netem*. Flow-level forwarding actions are enforced through the Ryu SDN controller. The application layer supports MQTT publish/subscribe traffic, HTTP/REST requests, and lightweight telemetry streams directed toward edge-hosted services. Packet counters, link statistics, queue occupancy, service delay, delivery ratio, resource usage, authentication events, and policy violations are aggregated every 5 seconds into a structured JSON-based MSD (see Fig.2). The main experimental parameters are reported in Table I.

```
  "selection_rationale": "string  // Short rationale (e.g., 'importance-ranked +
      constraint-satisfied')",
  "expected_performance": {
    "tpr": "number  // Expected true positive rate (if applicable)",
    "fpr": "number  // Expected false positive rate (if applicable)",
    "acc": "number  // Expected accuracy (if applicable)"
  },
  "estimated_cost": {
    "telemetry_bytes_s": "number  // Expected telemetry footprint (bytes per
        second) under this configuration",
    "cpu_ms": "number  //  Expected CPU time per request (ms)",
    "latency_ms": "number  // Expected added processing latency (ms)",
    "energy_mj": "number  // Expected energy cost per request (mJ)"
  }
},

"policy_hints": {
  "recommended_posture": "string  // High-level posture suggestion (e.g., light,
      balanced, strong-but-feasible)",
  "enforcement_actions": [
    {
      "action_type": "string  // Enforcement action (e.g., rate_limit, filter,
          isolate, steer_to_scrubber, ...)",
      "scope": "string  // Application scope (e.g., flow-class, node, segment,
          domain)",
      "trigger_conditions": "object  // Conditions on features/state that
          activate this action (thresholds/rules)",
    }
  ]
},
```

Fig. 2: Portion of the MSD artifact generated by the proposed setup, reporting elements from the *security* and *service* views.

The proposed framework is then compared against three reference strategies: (i) a *static* orchestration strategy where routes, service placement and telemetry rates are configured at deployment time and never changed during runtime; (ii) a *security-only* strategy, where anomalous nodes or flows are mitigated through isolation and access-control actions without considering their impact on service continuity; finally, (iii) a *service-only* strategy, where routing, priority and placement decisions are optimized to reduce delay and packet loss, while security information is used only for monitoring and does not constrain the admissible action space. Four attacks, each lasting between 100 and 180 seconds, are considered: *FLOOD*, an MQTT/HTTP volumetric attack (e.g., DoS and DDoS [18]) against constrained edge services; *STEALTH*, involving low-rate communications and scanning; *SLOW ACK*, which models subscriber or acknowledgment starvation; and *MIXED*, which alternates flood bursts, stealth flows, and slow-draining behaviors. Agents receive only the observations contained in the MSD, while attack labels are retained exclusively for offline evaluation. Three scenarios assess adaptability: *Standard*, with nominal traffic and no attacks; *Low Impact*, with each attack executed twice; and *High Impact*, with each attack repeated five times at shorter intervals to induce persistent stress.

TABLE I: Main experimental parameters

Parameter	Value
IoT devices	100–500
Device traffic rate	1–50 msg/s
Packet loss probability	0–10%
MSD update interval Δ	5 s
Historical window W	60 samples
Experiment duration	1 hour per replica
Independent repetitions	30 random seeds per configuration

B. Agentic Layer Implementation Details

This section provides the implementation details of the agentic layer of the proposed orchestration framework. The SPE-A is implemented through a two-stage security pipeline. First, a Kitsune-like ensemble of autoencoders is used to compute an anomaly score for each monitored flow since this model is suitable for online intrusion detection and can be trained on benign traffic without requiring a fully labeled attack dataset [19]. The computed anomaly scores are injected into a graph-based risk model implemented through GraphSAGE and GAT [20], [21], accounting for the fact that the impact of a suspicious device is not local but depends on its position in the communication graph, the services it reaches, and the paths it can influence. The SPA-A is implemented through a predictive service-assurance pipeline: A GNN-based SLA predictor estimates the probability of service violation from the current relation among flows, gateways, links, and edge functions [22] and, in parallel, Chronos-2 is used to forecast the short-term evolution of delay, packet loss, and queue occupation from the recent MSD history [23]. The RP-A is implemented as a PPO-based deep reinforcement learning policy trained offline and frozen during the evaluation [24], [25]. The policy observes the available edge resources, the current service placement, the predicted demand produced by SPA-A, and the risk constraints produced by SPE-A. Its objective is to select resource and placement actions that reduce service degradation while avoiding unnecessary migration and excessive control overhead. Finally, the OR-A is implemented as a hybrid decision module, in which Llama 3.1-8B-Instruct is used only to interpret the contextual policy information and to transform the reports produced by the other agents into structured candidate plans [26], and each candidate plan is passed to an OPA/Rego validator, which implements the secure-by-design constraints as hard rules [1].

[1]https://www.openpolicyagent.org/

This separation between neural interpretation and symbolic validation follows the neuro-symbolic design principle adopted in the framework in [27]. Plans violating at least one security, service, placement, or context constraint are discarded before optimization, while the final orchestration action is selected only among the validated plans as discussed in Section III.

V. Performance Evaluation and Results

The evaluation considers four metrics: *average application-level delay*, *packet loss*, *security violation ratio*, and *additional control overhead*. They respectively assess service responsiveness, message delivery reliability, the percentage of malicious or policy-incompatible interactions not prevented or mitigated, and the extra traffic generated by monitoring, agent coordination, and enforcement. Under nominal traffic and without attacks, the proposed framework achieves the lowest delay and packet loss, reducing the average delay by 98.6%, from 24411 ms with the *static* strategy to 337 ms. Unlike the *security-only* and *service-only* approaches, which optimize individual dimensions, the proposal jointly minimizes delay, packet loss, and security violations through coordinated agentic orchestration. When attacks are introduced under nominal traffic, even in the *low impact* scenario, the limitations of single-dimension strategies become more evident. The *static* configuration suffers from a sharp degradation, with an average delay of 56632 ms, packet loss equal to 12.99%, and a security violation ratio of 45.76%, while the *security-only* strategy significantly reduces the security violation ratio to 6.23%, confirming that direct mitigation actions are effective against malicious behavior; however, the delay remains high for the *security-only strategy*, since it does not reason on the service impact of security actions. The *service-only* strategy reduces delay and packet loss compared with the *static* and *security-only* approaches, reaching 20780 ms and 5.96%, respectively. By redistributing traffic and reducing overload, it partially mitigates attacks that cause resource pressure, queue growth, congestion, and SLA degradation. Nevertheless, the security violation ratio remains high at 33.21%, showing that service optimization alone cannot distinguish between legitimate stress and malicious or policy-incompatible behavior. In the same scenario, the proposal reduces the delay to 1753 ms and the packet loss to 0.58%, while also limiting the security violation ratio to 2.51%, confirming that the advantage of the proposed approach is not only the presence of multiple control actions, but the fact that these actions are selected through a shared system state and filtered by security-aware admissibility constraints. The critical attack condition further highlights the role of context-aware coordination: the *static* strategy becomes unable to preserve both service continuity and security, reaching 61051 ms of average delay, 51.58% packet loss, and 77.32% security violation ratio, suffering from the high intensity with which the different attacks are performed. The *security-only* strategy keeps the security violation ratio considerably lower than the *static* and *service-only* strategies, but its delay and packet loss remain high, with 38599 ms and 12.81%, respectively, indicating that mitigation without service-aware adaptation may protect the system from part of the malicious behavior while still leaving the infrastructure in a degraded operational state. The *service-only* strategy behaves oppositely: it reduces the delay with respect to the *static* and the *security-only* strategies, but it cannot prevent security violations, which remain equal to 65.94%. The proposed framework maintains the average delay at 1802 ms, limits packet loss to 2.42%, and reduces the security violation ratio to 6.88%. Although the security violation ratio increases with respect to the *low intensity* attack condition, the increase is consistent with the higher attack frequency and reduced recovery time between malicious phases; however, the proposal can preserve both service availability and security compliance under the most adverse condition. The proposed framework introduces monitoring, coordination, and enforcement overhead, but remains comparable to the *security-only* strategy, differing by only 0.35% under critical conditions. This cost is justified by a 95.3% reduction in packet loss over the *static* strategy and an 89.6% reduction in security violations over *service-only* orchestration, demonstrating the effectiveness of holistic and adaptive closed-loop control.

VI. Conclusions

This work addressed the challenge of a holistic, adaptive, and real-time orchestration and management of security and network performance under heterogeneous and dynamic IoT environments. The proposed orchestration framework leverages the agentic AI paradigm, deploying role-specific agents to materialize a Multi-View State Descriptor (MSD) that comprehensively synthesizes device, network, security, and contextual information of the IoT environment. An Orchestrator Agent (OR-A) uses the MSD to trigger a decision-making control loop that, based on the gathered observations and system evolution, computes the best security and network management plan to apply to the underlying network. A comprehensive experimental campaign proves the benefits of the proposal against *static* and *single* target KPI optimization, namely *service-only* and *security-only*, reducing delay, packet loss, and security violations with limited overhead.

Acknowledgments

This work was partially supported by Project PON SUNS - Secure Ubiquitous Networks and Services financed by MISE, (Azione 1.1.2) – CUP B29H26000360007. Period:2026-2028.

References

[1] A. F. Santamaria *et al.*, "A real IoT device deployment for e-Health applications under lightweight communication protocols, activity classifier and edge data filtering," *Comput. Commun.*, vol. 128, pp. 60–73, Sep. 2018.

[2] F. Guo *et al.*, "Enabling Massive IoT Toward 6G: A Comprehensive Survey," *IEEE IoT J.*, vol. 8, no. 15, pp. 11 891–11 915, Mar. 2021.

[3] M. G. Spina *et al.*, "Optimal in-network distribution of learning functions for a secure-by-design programmable data plane of next-generation networks," *Comput. Networks*, vol. 282, p. 112281, Jun. 2026.

[4] A. Mokhtari and A. Ksentini, "Hierarchical Multi-Agent RL TE in SD-WAN based Cloud Edge Continuum Interconnection," in *2025 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. IEEE, 2025, pp. 27–31.

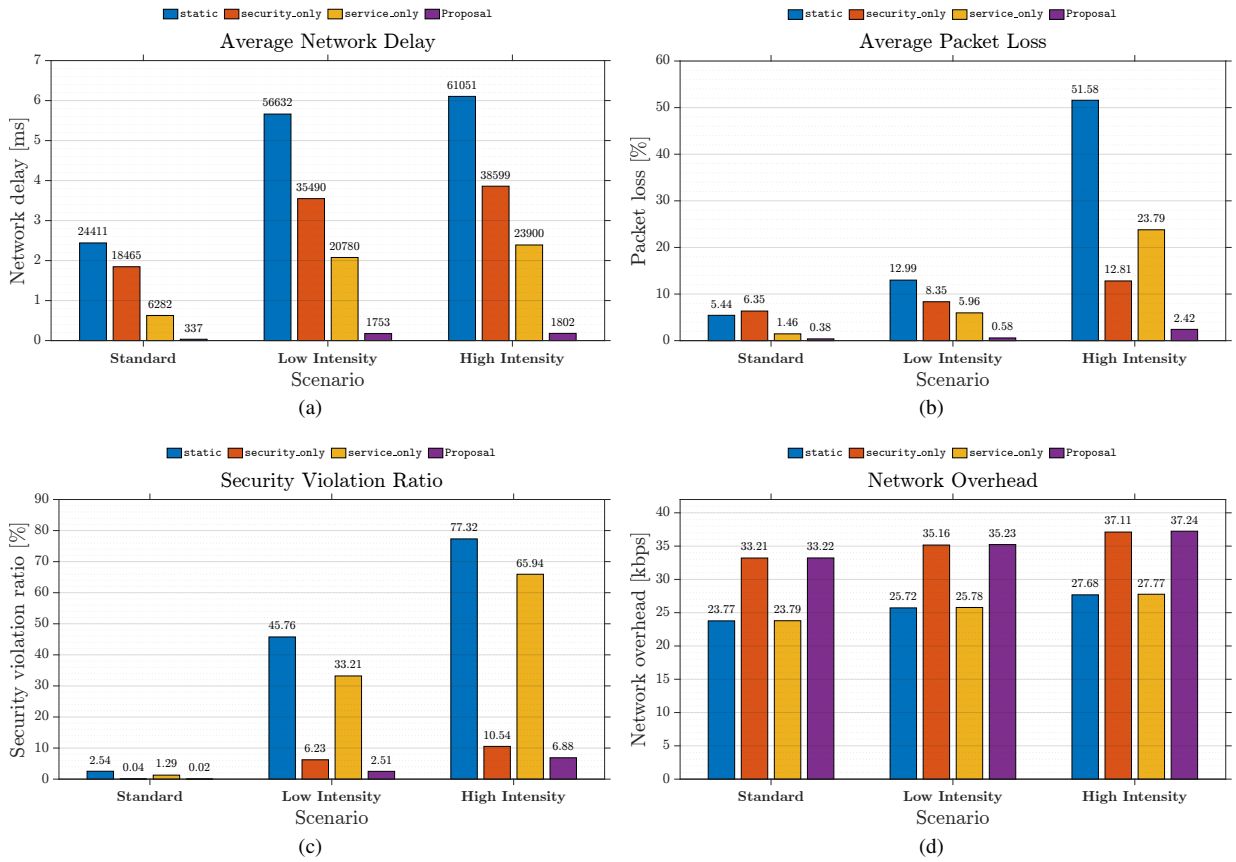


Fig. 3: Results related to (a) Average Network Delay, (b) Average Packet Loss, (c) Average Security Violation Ratio, and (d) Average Network Overhead.

[5] H. F. Shahid *et al.*, "IoT Service Orchestration in Edge–Cloud Continuum With 6G: A Review," *IEEE IoT J.*, vol. 13, no. 10, pp. 20 339–20 358, Feb. 2026.

[6] X. Chen *et al.*, "Toward 6G Native-AI Network: Foundation Model-Based Cloud-Edge-End Collaboration Framework," *IEEE Commun. Mag.*, vol. 63, no. 8, pp. 23–30, Jul. 2025.

[7] X. Li *et al.*, "The Agentic-AI Core: An AI-Empowered, Mission-Oriented Core Network for Next-Generation Mobile Telecommunications," *Engineering*, vol. 56, pp. 104–119, Jan. 2026.

[8] A. Braeken *et al.*, "6G AI Security: From Fundamentals to Offensive and Defensive Landscape in 6G," *IEEE Commun. Surv. Tutorials*, vol. 28, pp. 4765–4799, Jan. 2026.

[9] S. Batewela *et al.*, "Addressing Security Orchestration Challenges in Next-Generation Networks: A Comprehensive Overview," *IEEE Open J. Comput. Soc.*, vol. 6, pp. 669–687, Apr. 2025.

[10] M. G. Spina, A. Boukerche, and F. De Rango, "An IoE-Powered Framework for Adaptive Energy-Security Trade-Off in IoT," *IEEE Network*, vol. 40, no. 2, pp. 65–71, Dec. 2025.

[11] D. Cajaraville-Aboy *et al.*, "Decentralized Orchestration Framework for Distributed AI Deployments across Fluid Computing Environments," in *2025 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. IEEE, 2025, pp. 27–31.

[12] A. F. Gentile *et al.*, "A VPN Performances Analysis of Constrained Hardware Open Source Infrastructure Deploy in IoT Environment," *Future Internet*, vol. 14, no. 9, p. 264, Sep. 2022.

[13] M. R. H. Mahin *et al.*, "Secured and Standardized Intelligent Zero-Touch 6G Framework for Edge-AI Applications," *IEEE Commun. Stand. Mag.*, pp. 1–5, Feb. 2026.

[14] A. J. A. Majumder *et al.*, "Advancing Intelligent Energy Harvesting in IoT Networks Using Agentic and GenAI Approaches," in *2025 28th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2025, pp. 19–21.

[15] H. Li *et al.*, "Future Factories With 6G: Agentic AI and Cyber–Physical Digital Twins," *IEEE IoT J.*, vol. 13, no. 9, pp. 17 990–18 006, Oct. 2025.

[16] A. Antonopoulos *et al.*, "AGILE-6G: Agentic AI for Autonomous Management of 6G Network/Application Services," *IEEE Network*, pp. 1–9, Dec. 2025.

[17] M. Elkael *et al.*, "AgentRAN: An Agentic AI Architecture for Autonomous Control of Open 6G Networks," *arXiv*, Aug. 2025.

[18] G. Potrino *et al.*, "Modeling and evaluation of a new IoT security system for mitigating DoS attacks to the MQTT broker," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 15–18.

[19] Y. Mirsky *et al.*, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," *arXiv*, Feb. 2018.

[20] W. L. Hamilton *et al.*, "Inductive Representation Learning on Large Graphs," in *Advances in Neural Information Processing Systems*, 2017.

[21] P. Veličković *et al.*, "Graph Attention Networks," *arXiv*, Oct. 2017.

[22] A. Maroudis *et al.*, "Leveraging Graph Neural Networks for SLA Violation Prediction in Cloud Computing," *IEEE Transactions on Network and Service Management*, pp. 1–1, 01 2023.

[23] A. F. Ansari *et al.*, "Chronos-2: From Univariate to Universal Forecasting," *arXiv*, Oct. 2025.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, Jul. 2017.

[25] A. Raffin *et al.*, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[26] A. Grattafiori *et al.*, "The Llama 3 Herd of Models," *arXiv*, Jul. 2024.

[27] G. Marra *et al.*, "From statistical relational to neurosymbolic artificial intelligence: A survey," *Artificial Intelligence*, vol. 328, p. 104062, 2024.