Edge Sandbox: A Resource -Aware Intelligent Architecture for Autonomous Networks

L L >Q+ TSE>MM>

Content

6G based AI

The high-level architectural framework of AN, includes several main interacting subsystems:

a) Autonomy engine

This is the core innovation engine, responsible for generating and validating new autonomous controllers. It includes the exploratory evolution subsystem to create or adjust controller designs, and the experimentation subsystem to validate these controllers in an autonomous network sandbox environment.

a) Dynamic adaptation subsystem

Responsible for integrating and operating validated controllers into the operating underlying network, using adaptation controllers to manage autonomous applications during runtime.

a) Knowledge base - KB system

A central component that manages the lifecycle of "knowledge" (analyzed data and information) in AN, storing controller specifications, models, metrics, and learned insights.

a) AN orchestrator

Manages overall workflows and processes within the AN framework itself.

a) E2E network orchestrator

Manages and orchestrates network services and resources in the underlying network.

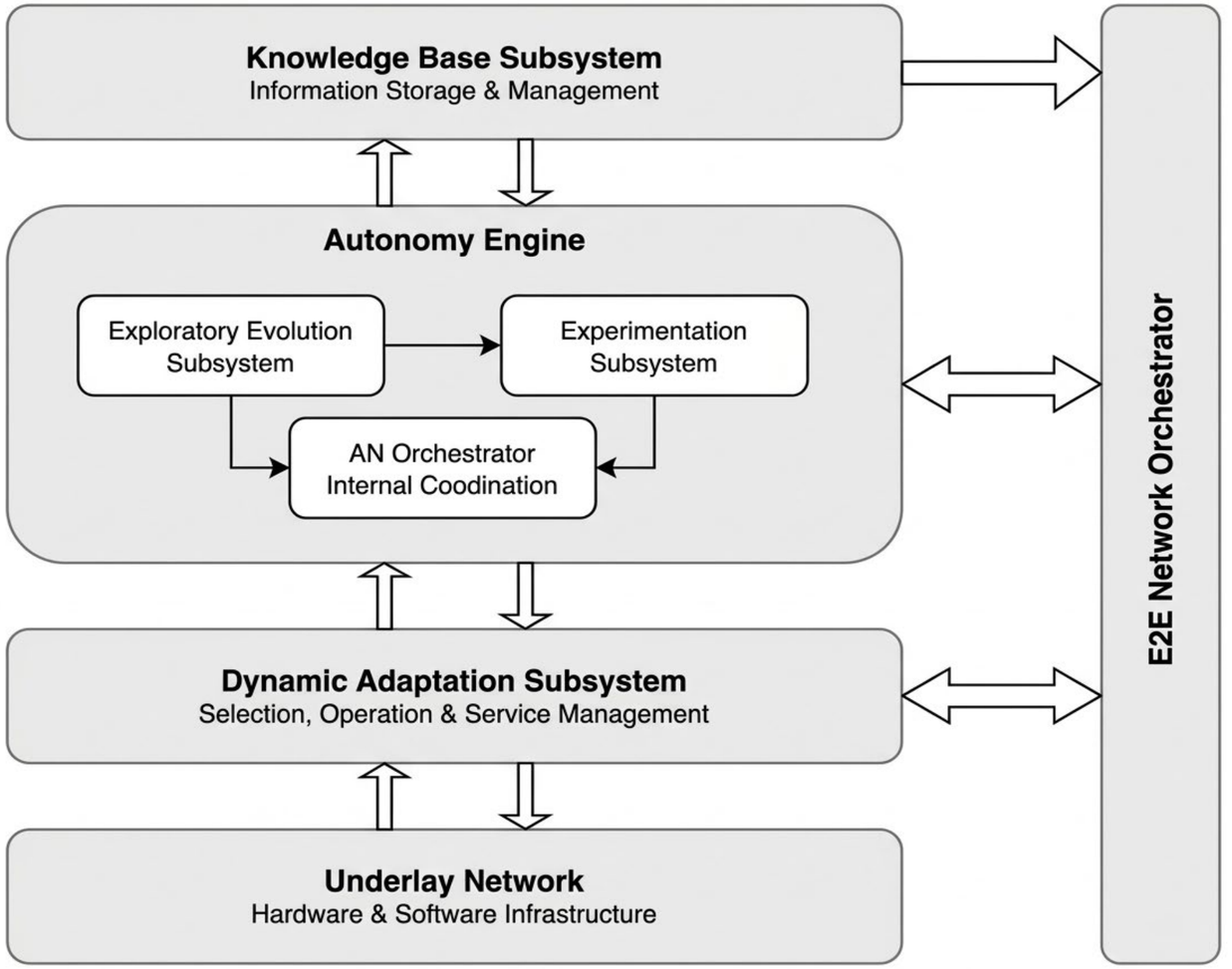
a) Underlay network

The actual network infrastructure that AN manages.

Autonomous networks - networks that possess inherent capabilities to monitor their own state, self-operate, self-recover from failures, self-patch vulnerabilities, self-protect against threats, self-optimize performance, and self-reconfigure their structure as well as functions to respond to changes in environmental conditions or operational objectives .

ITU-T Recommendation Y.3061

Network design diagram



ITU-T Recommendation Y.3061

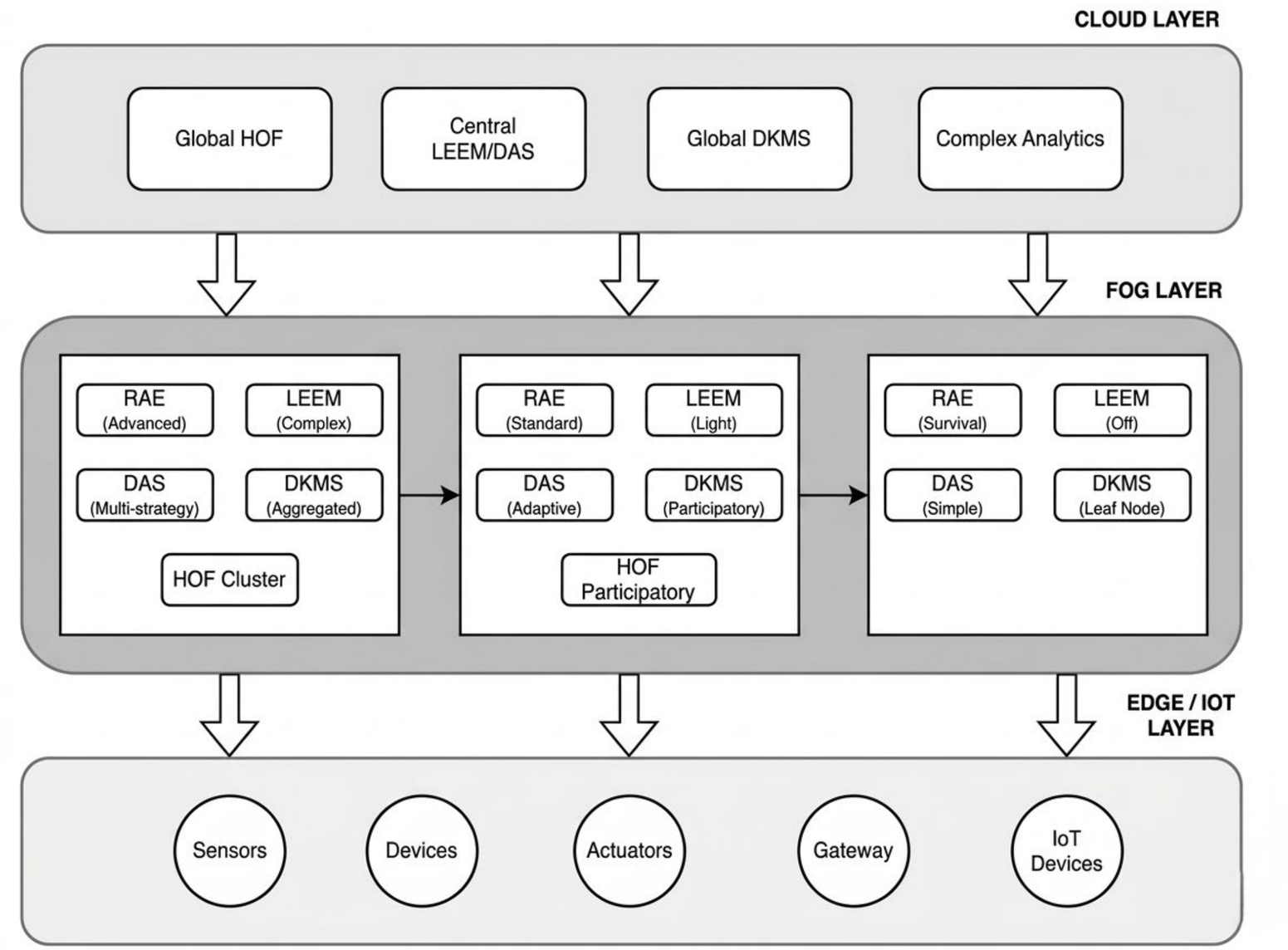
Comparison of cloud computing, fog computing, and edge computing models

Attribute	Cloud computing	Fog computing	Edge computing
Primary processing location	Centralized, remote data centers	Near users/devices, an intermediate layer between edge and cloud	Directly on devices or gateways very close to devices
Typical latency	Higher (tens of milliseconds to seconds)	Low to medium (milliseconds)	Very low (under milliseconds to milliseconds)
Scalability model	Very high, on demand	Horizontally scalable by adding Fog nodes, less centralized than the cloud	Limited by the resources of edge devices or local edge clusters
Local data processing volume	Minimal	Medium to high (filtering, aggregation, local analysis)	Low to medium (specialized processing for devices)
Use case examples	Big data analytics, long-term storage, and large-scale web applications	Smart cities, industrial IoT, autonomous vehicles, remote healthcare, content distribution (CDN)	Real-time device control, instant sensor response, on-device AR/VR
Security/Privacy focus	Centralized data center protection, data transmission encryption	Local data processing with sensitive data, access control at fog, and reduced data transmission to the cloud	On-device protection, limited data processing at source
Key advantages	Large-scale expandability, on-demand resource provision, and cost efficiency for large processing	Real-time recognition capability, reduced bandwidth, and local recovery capability	Extremely low latency, offline operation, reduced data transmission
Main challenges	Latency, network connectivity dependence, and large data transmission costs	Complex management, heterogeneity, standardization, and distributed security	Limited resources, limited processing capability, and distributed device management
Relationship with other layers	Top layer provides centralized resources	Intermediate layer connects the edge and cloud, extending cloud capabilities to the edge	Bottom layer, closest to the data source

Comparative analysis of architecture with advanced autonomous fog architectures

Feature	Proposed architecture	Open Fog (IEEE 1934)	ETSI MEC	FRATO	General Autonomic Management (MAPE-K)
Primary focus	Deep, hierarchical, resource-aware autonomy for heterogeneous fog nodes	Comprehensive reference architecture, interoperability	Cloud resources at the mobile edge, low latency for mobile applications	Adaptive task partitioning, resource-aware for task placement	General self-management loop
Autonomy support	Comprehensive (Self-CHOP through RAE, LEEM, DAS, HOF)	Encouraged, implementation details open	Focus on resource orchestration, less on node autonomy	Autonomy in task partitioning decisions	Variable, often assumes sufficient resources
Resource-aware mechanisms	RAE (adjusts complexity of autonomous agents), LEEM (resource-aware evolution), DAS (resource-aware deployment)	Resource management mentioned, less detailed than HAFA	Resource management for edge applications	Partitioning policy selection based on available fog resources	Often implicit or application-specific
Hierarchical model	Yes, 3-tier (cloud, fog, edge/IoT) with hierarchical orchestration (HOF)	Yes (Hierarchy pillar)	Multi-server edge possible, less focus on deep hierarchy	IoT-Fog-Cloud model, focus on inter-tier partitioning	Not inherent, can be layered
Knowledge management	DKMS (hierarchical, compressed, selective synchronization using Merkle trees)	Not specified in detail	Not a primary focus	Not specified in detail	Knowledge Base (K) is central, but distribution varies
On-Node evolution	LEEM (lightweight evolution, safe on-node experimentation)	Not specified in detail	Not a primary focus	No controller evolution	Often offline or cloud-based training
Target node constraints	From severely constrained to powerful	General fog nodes	Edge servers (typically well-resourced)	Heterogeneous fog nodes, does not emphasize extreme low constraints like HAFA	Often assumes medium to high resources
Validation approach	Formal analysis and extended experimental validation	Conceptual, focus on interoperability	Standards-oriented, focus on service APIs	Simulation-based for partitioning performance	Varies by implementation

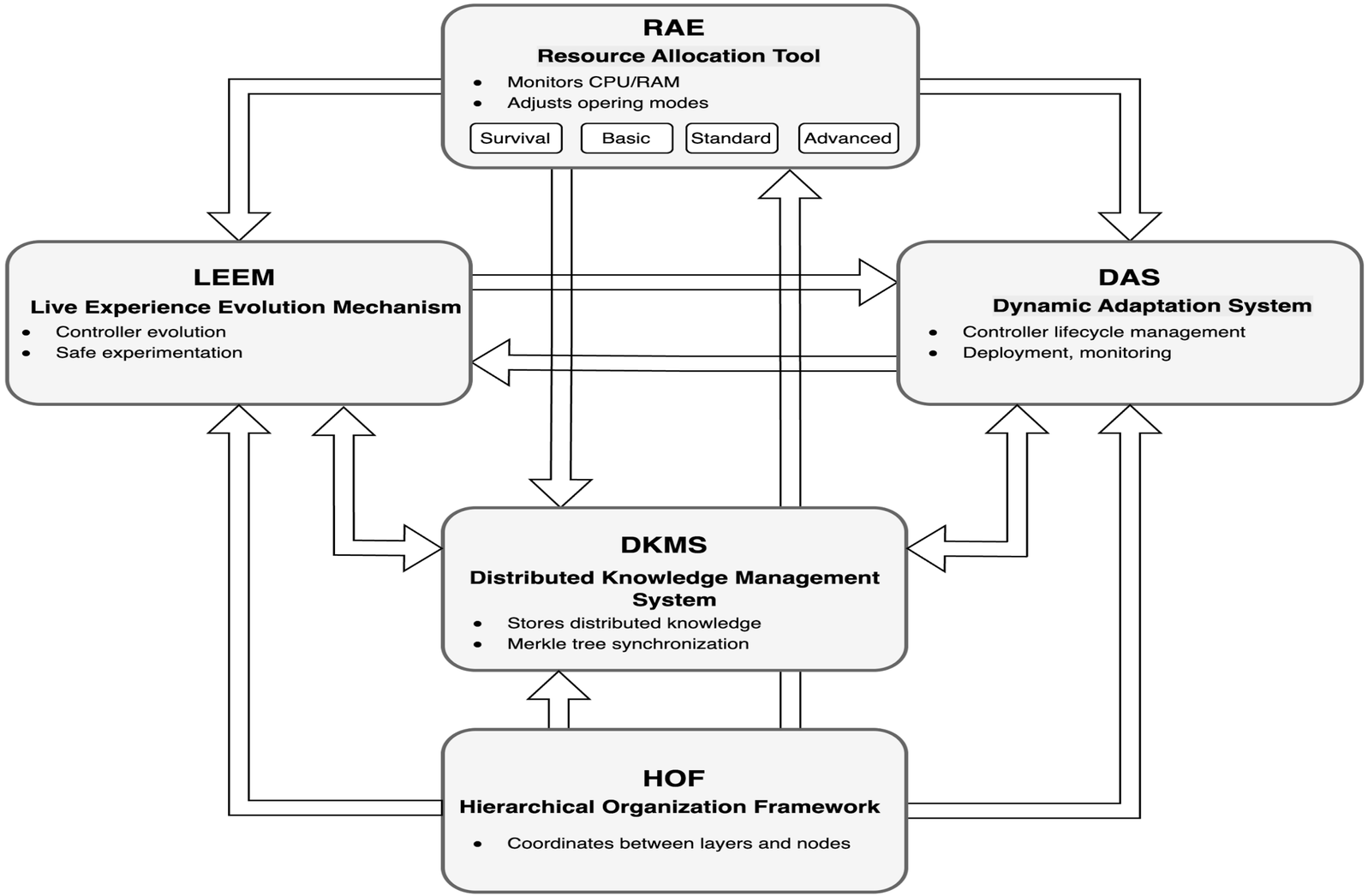
Architecture and components



Distribution of functions and responsibilities of the architecture component by the architectural tier

Architectural Tier	Primary proposed architecture Components/Functions	Description of Main Responsibilities/Activities at Tier/Class
Cloud tier	• HOF global • LEEM/DAS central • DKMS global	• Policy establishment and global objectives • Inter-cluster coordination • Complex ML model training • Storage of global progressive templates • Central DKMS repository/backup storage
Fog tier (Class III - well-resourced)	• HOF cluster • RAE (advanced mode) • LEEM (complex) • DAS (multi-strategy) • DKMS (cluster aggregation)	• Cluster coordination • Complex autonomous decision-making • Full controller evolution (NSGA-II) • Comprehensive experimentation • Knowledge aggregation and strategic dissemination/expansion
Fog tier (Class II - moderately constrained)	• HOF participation • RAE (standard mode) • LEEM (lightweight) • DAS (adaptive component) • DKMS (cluster participation)	• Cluster coordination participation • Moderately complex autonomous decision-making • Lightweight controller evolution (µGA) • Experimentation based on templates/canary • Strategic knowledge sharing/reception.
Fog tier (Class I - severely constrained)	• RAE (survival/basic mode) • DAS (simple behavior) • DKMS (leaf node)	• Minimal resource monitoring • Basic rule-based behavior implementation • Simple controller execution • Core knowledge storage and synchronization at a minimal level
Edge/IoT tier	(No direct architecture components)	• Primary data source • Object receiving control commands from Class I/II fog nodes • Dependent on fog tier for intelligent processing and decision-making

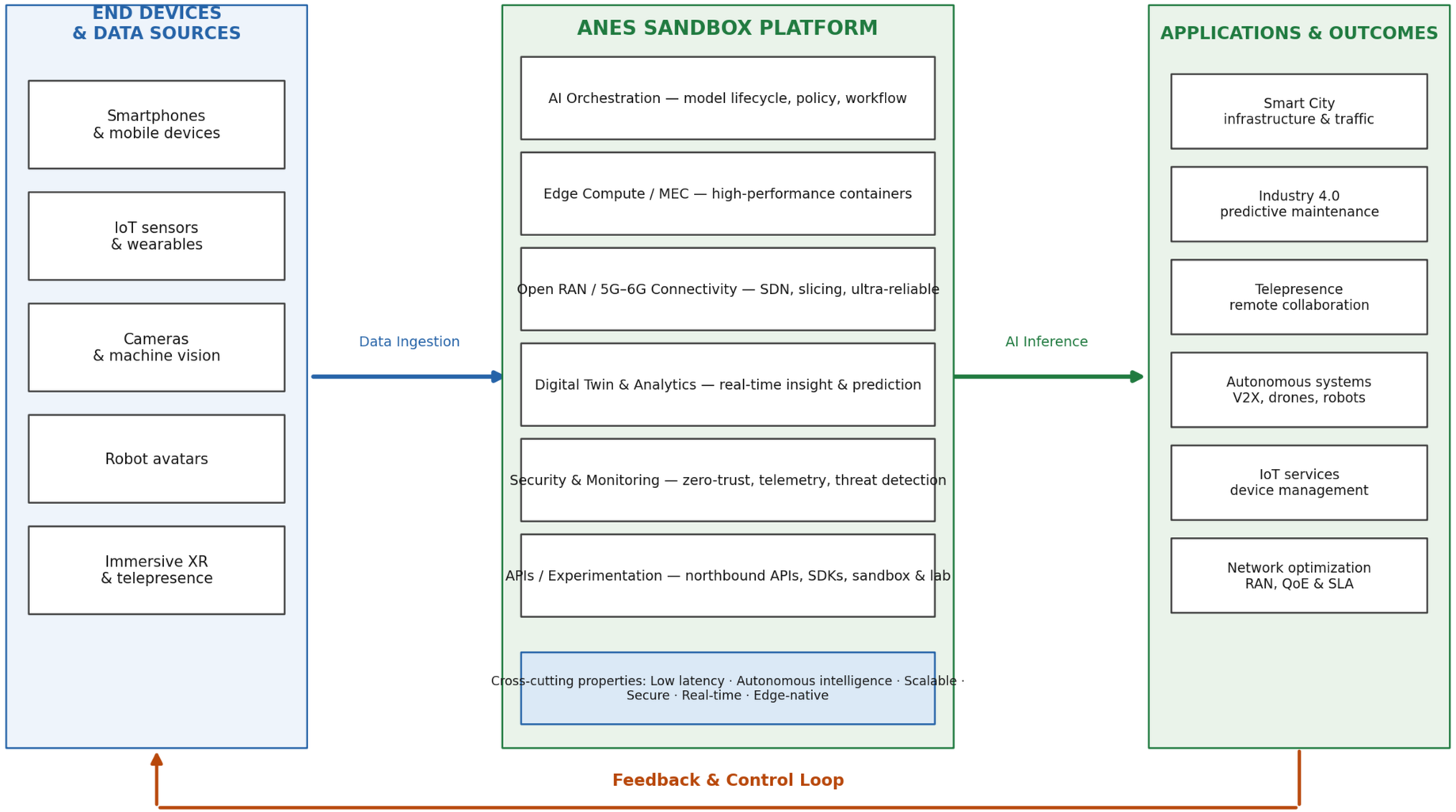
Interaction model between architecture's fundamental components



Node operating modes and detailed information

Mode	Resource Threshold	Autonomous Capabilities	Algorithms and Techniques	Activity Objectives
Survival	RAM < 5-10% CPU < 5%	• Minimal system monitoring • Critical application only • Critical error reporting • No learning capability	• Simple FSM • Lookup tables • Hard-coded If-Then rules	• Maintain basic operation • Prevent system collapse • Report critical states to HOF
Basic	RAM 10-20% CPU 5-15%	• Linear reference control • Simple pattern detection • Local optimization with constraints • Basic elasticity learning	• EWMA • Lightweight PA • Simple SSD • Simple heuristics	• Optimize local controllers • Early problem detection • Basic reference participation
Standard	RAM 20-40% CPU 15-25%	• Multi-level adaptation • Moderate ML complexity inference • Dynamic strategy selection • Rich DAS interaction	• Decision trees • TinyML • Small NN • Strategy selection	• Flexible adaptation • Performance/resource balance • Multi-application support
Advanced	RAM > 40% CPU > 25%	• LEEM evolution initiation • Sandbox experimentation • Multi-objective optimization • Deep and complex RL	• NSGA-II • RL • Complex GA • Virtual sandboxing	• Comprehensive optimization • New strategy exploration • Complete autonomy

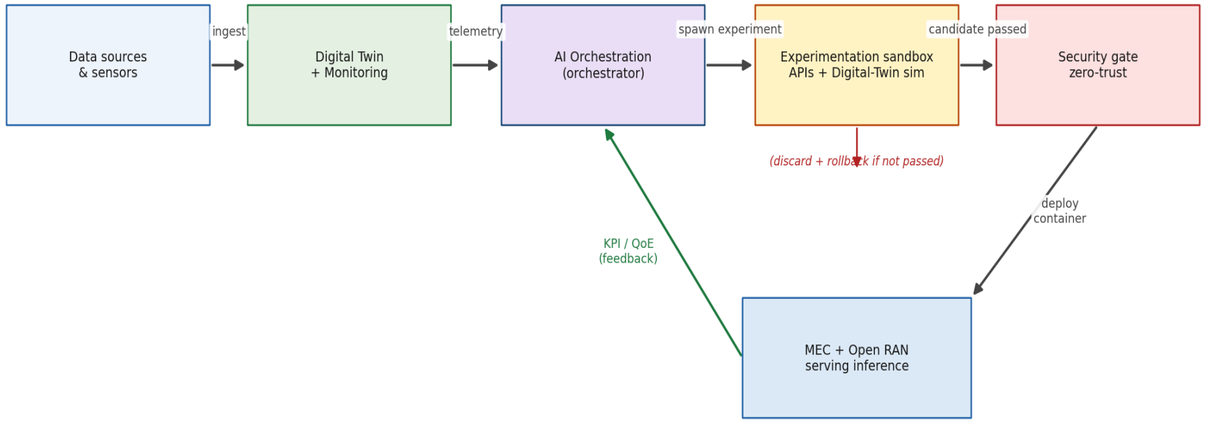
AI-Native Edge Sandbox Architecture



Components and operating principle

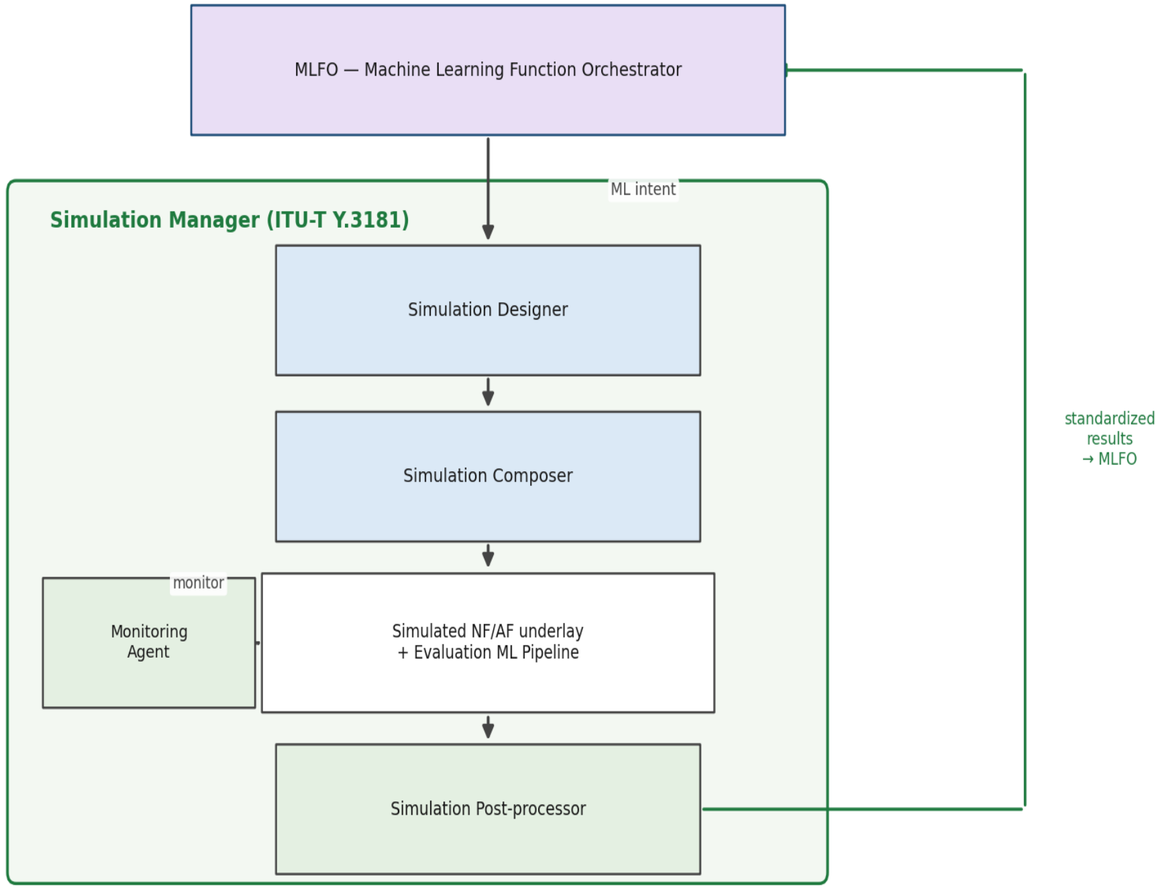
The role and operating principle of each component are as follows:

- **AI Orchestration —** acts as the orchestration brain: it manages the full controller lifecycle (generate — test — promote — operate), interprets application intent into policy, and makes global decisions.
- **Edge Compute / MEC —** provides high-performance, containerized compute at the edge; it packages controllers into containers and serves inference, supporting non-disruptive hot-swap.
- **Open RAN / 5G–6G Connectivity —** provides connectivity and network slicing over Open RAN/SDN, guaranteeing latency and reliability per the service-level agreement (SLA) for each application flow.
- **Digital Twin & Analytics —** maintains a real-time digital twin; it is both a source of insight and prediction and a safe simulation environment for testing controllers before real deployment.
- **Security & Monitoring —** enforces the zero-trust principle: it checks integrity, policy compliance and threat level of a candidate controller before allowing promotion; it monitors telemetry continuously.
- **APIs / Experimentation —** provides the environment and tools to trial controllers (sandbox, lab) together with northbound APIs and SDKs, realizing the experimentation phase of the evolution loop.



Mapping of sandbox components to ITU-T Y.3181

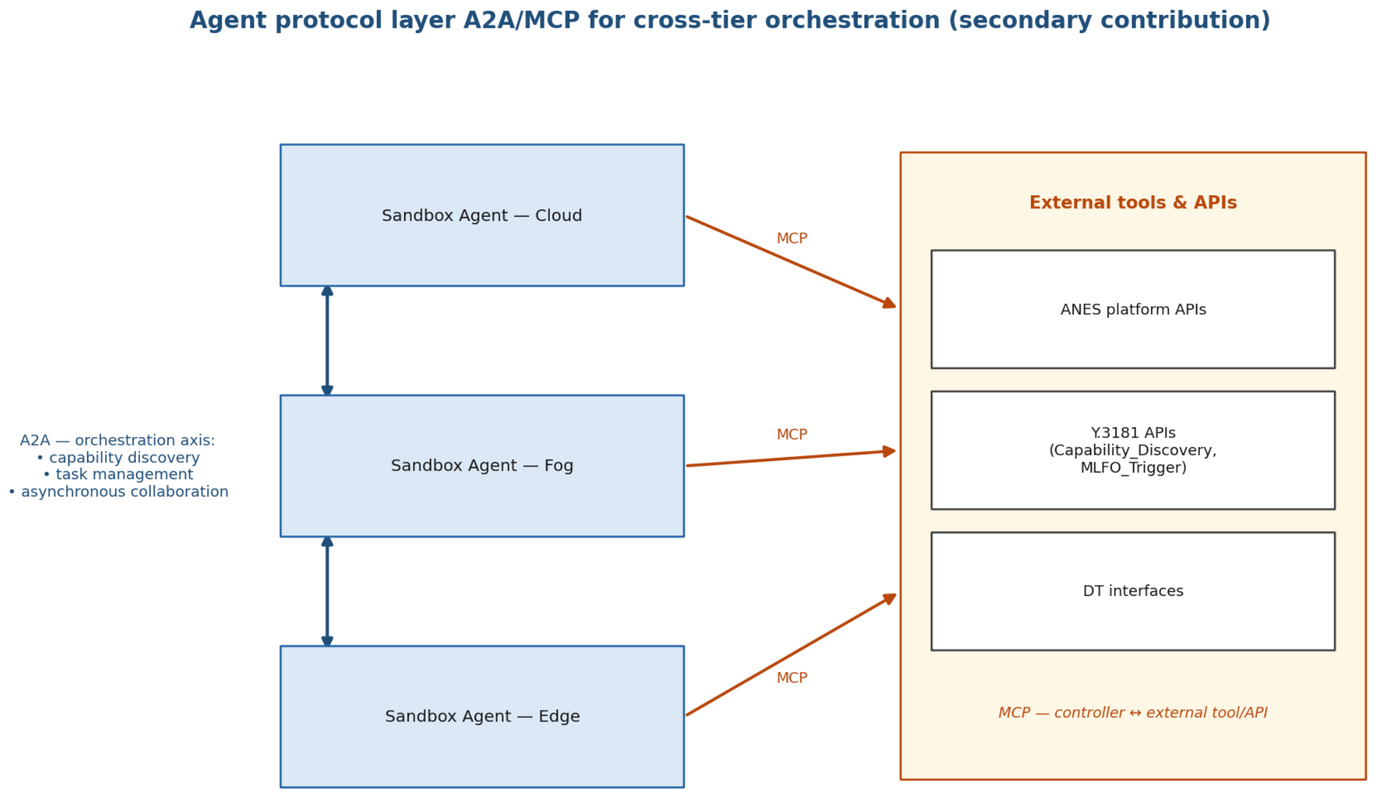
- **DT Resolver —** receives the what-if query from the cross-tier orchestrator, analyses the request and determines the scope of the digital twin needed.
- **DT Instantiator —** creates a separate instance for each query, ensuring that concurrent test sessions do not conflict.
- **DT Executor —** combines multiple DT modules and simulators (for example a RAN DT module, a city-mobility simulator, a rendering DT module), runs them in parallel and merges the results for multi-domain what-if analysis — each simulator provides information the others cannot determine on their own.
- **DT Repository —** stores versioned virtual models, ready for the Executor to load on demand.



Evaluation ML Pipeline runs on the simulated underlay (not the live network); robustness testing per REQ-ML-SANDBOX-015.

Agent protocol layer A2A/MCP for cross-tier orchestration

- **A2A (Agent-to-Agent) —** used for communication among sandbox agents across tiers (Cloud–Fog–Edge): capability discovery, task management and asynchronous collaboration. This is the orchestration axis of the system.
- **MCP (Model Context Protocol) —** used for communication between controllers and external tools/APIs, including the ANES platform APIs, the Y.3181 APIs and the Digital Twin interfaces. This is the tool-access axis.



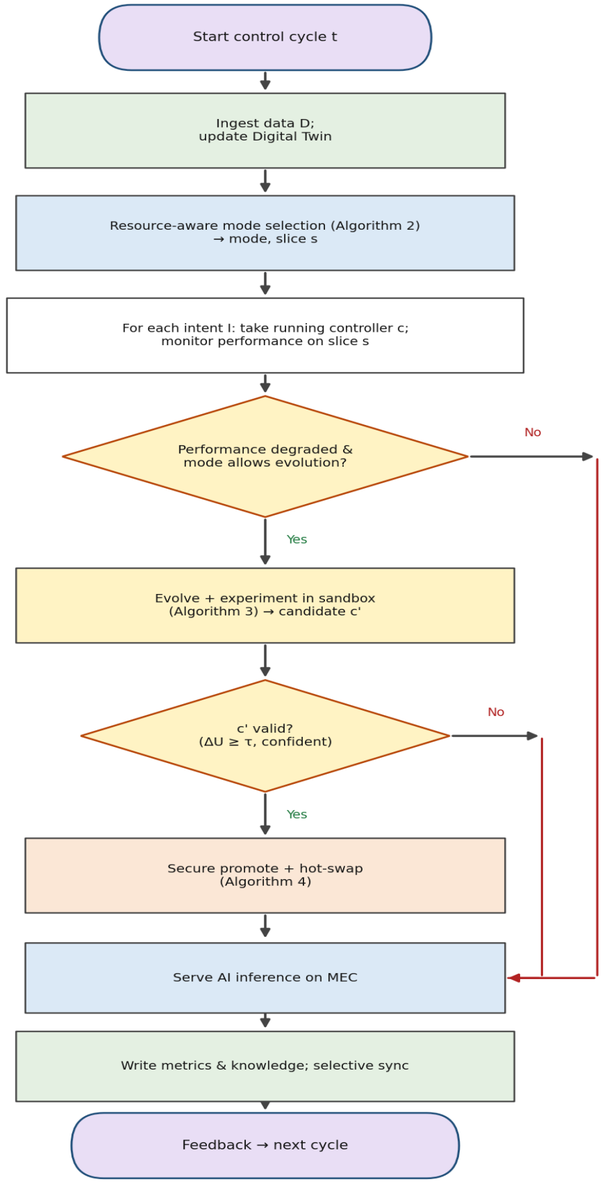
Operating algorithms of AI-Native Edge Sandbox Architecture

The operation of AI-Native Edge Sandbox Architecture is described by :

1. Master algorithm
   - Master control and feedback loop
1. Three sub-procedures (Algorithms)
   - Mode selection and slice allocation
   - Evolution and experimentation in the sandbox
   - Secure promotion (zero-trust) and hot-swap

Operating algorithms of AI-Native Edge Sandbox Architecture

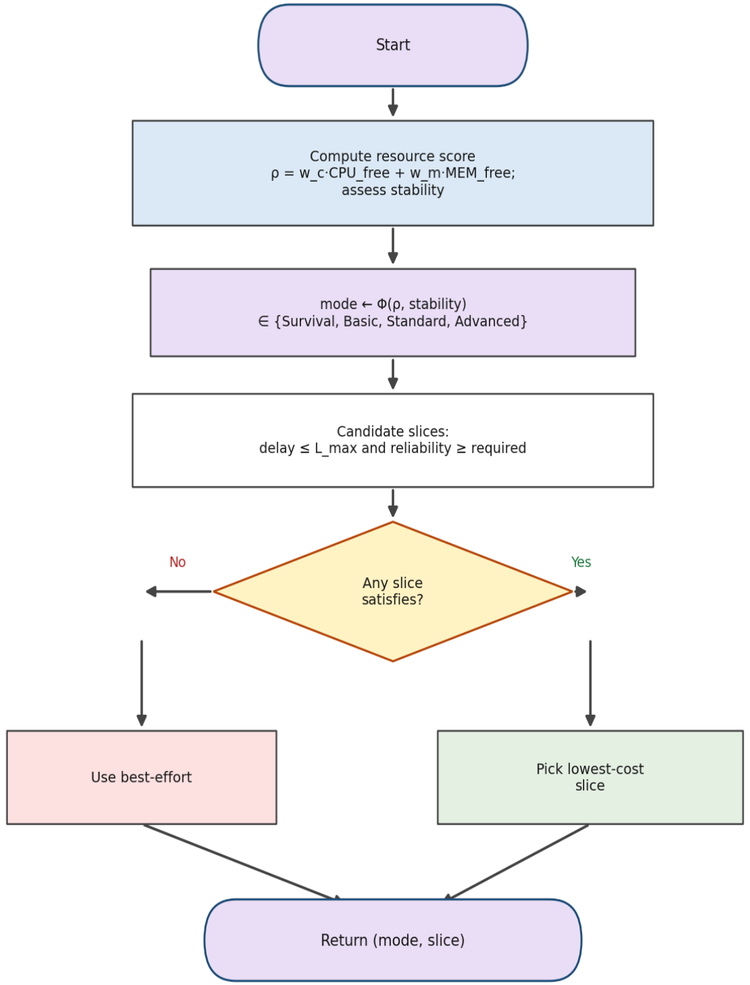
This Algorithm is the platform's feedback and control loop: each cycle, it updates the digital twin, selects mode and slice, monitors the running controller, triggers evolution when needed, promotes a qualifying candidate, serves inference and writes knowledge.



Operating algorithms of AI-Native Edge Sandbox Architecture

Mode selection and slice allocation:

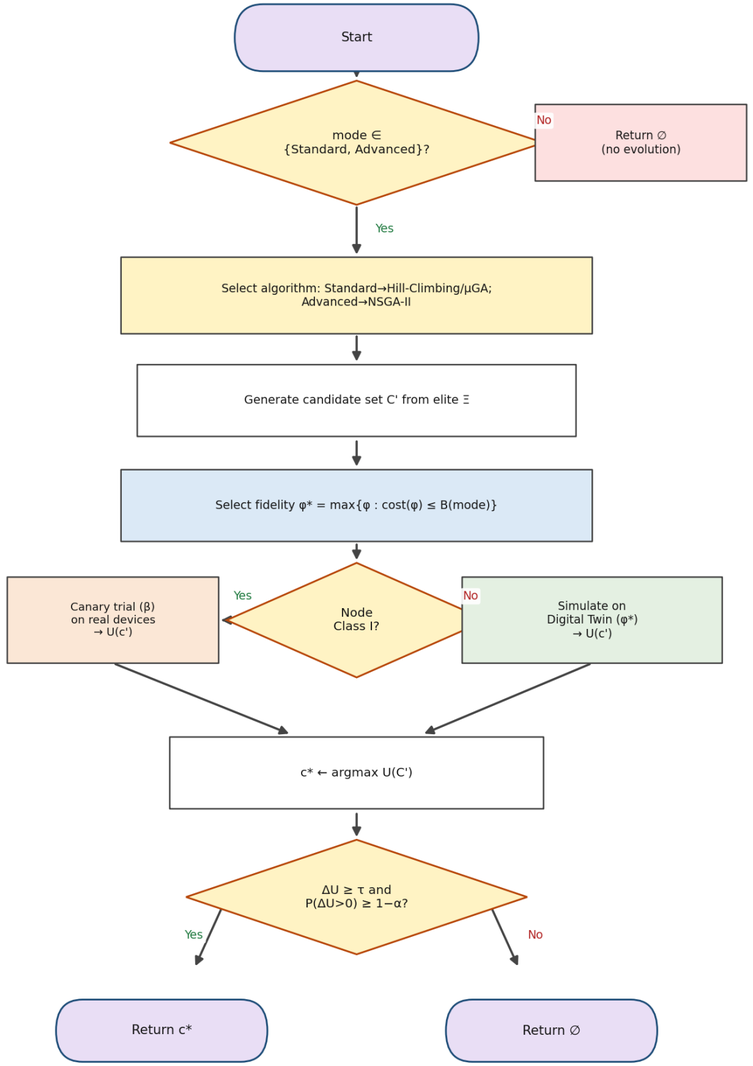
This procedure computes the resource score to choose the operating mode, then selects a network slice matching the intent's SLA.



Operating algorithms of AI-Native Edge Sandbox Architecture

Evolution and experimentation in the sandbox:

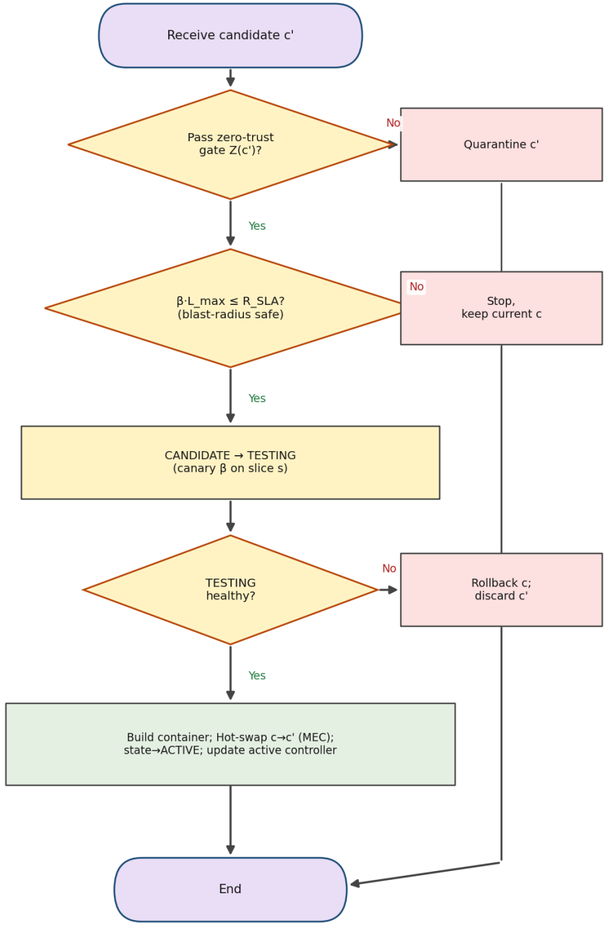
This procedure generates candidate controllers and evaluates them safely by simulation on the query-driven Digital Twin for powerful nodes, or by canary testing on constrained nodes, then returns the best candidate if it meets the promotion criterion



Operating algorithms of AI-Native Edge Sandbox Architecture

Secure promotion (zero-trust) and hot-swap:

This procedure applies the security gate and blast-radius control before putting a new controller into operation, with rollback if the testing phase reveals a problem.



MOBILE EDGE COMPUTING · 6G · MULTI-AGENT REINFORCEMENT LEARNING

Integrating Spatiotemporal Prediction with Cooperative Multi-Agent Reinforcement Learning

2 Motivation

OPEN CHALLENGES

Resource contention, no coordination

With only local views, many BSs offload to the same edge server, so its shared queue overloads and tasks miss deadlines near saturation.

Reactive, no foresight

Decisions rely on the instantaneous state alone; agents react only after congestion, leaving spatiotemporal predictive signals unused.

Unrealistic workloads

Fixed-probability or random task generation ignores the burstiness and spatial structure of real backbone network traffic.

OUR CONTRIBUTIONS

Real-traffic two-tier MEC model

FIFO + GPS queues, three-part energy, deadline-based delay and a drain phase — all driven by Abilene OD matrices.

ConvLSTM spatiotemporal predictor

An encoder–decoder forecasts the OD matrix; the predicted size $\hat{\lambda}_m(t+1)$ is added to each BS observation.

MAPPO under CTDE (Dec-POMDP)

Centralized critic, action mask, delayed reward and energy-conserving per-agent credit assignment.

Comprehensive evaluation

Against four baselines, four traffic loads and five seeds, on delay, drop rate and energy per successful task.

3 System Model: Two-Tier edge Architecture

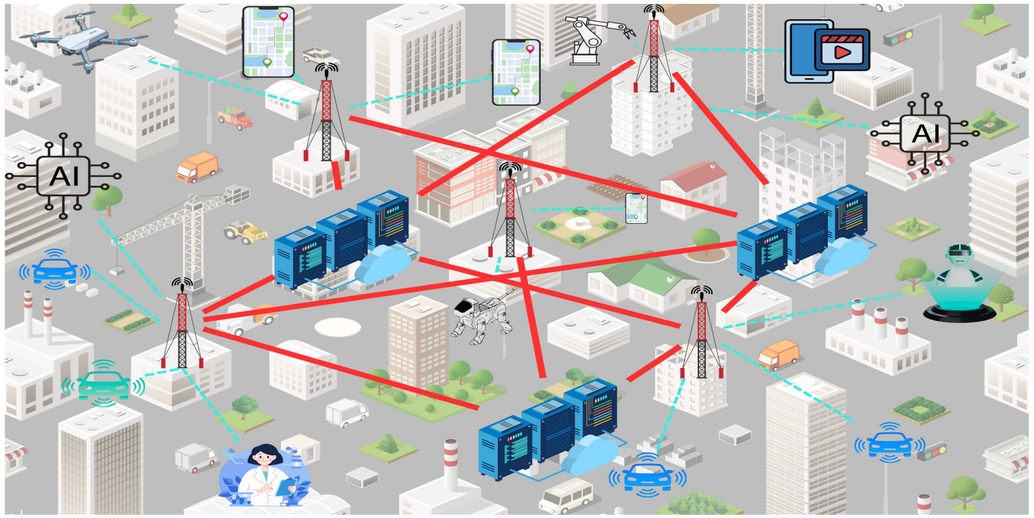


Fig. 1 — BSs offload tasks to shared MEC servers over wired links; UEs attach to BSs wirelessly.

The system has two tiers — M = 12 base stations (BSs) and N = 5 edge servers — over synchronized time slots. Each BS either computes a task locally or offloads it over a wired link to one edge server. Every BS keeps two FIFO queues (compute + transmission); each MEC server shares its CPU among active queues by Generalized Processor Sharing. Task sizes come from real Abilene OD traffic, preserving realistic burstiness.

FORMAL MODEL · M = 12 BS, N = 5 MEC (Abilene)

Traffic → task mapping

$$\lambda_m(t) = \begin{cases} 0 & V_m(t) < \epsilon V_m^{\max} \\ \dfrac{V_m(t)}{V_m^{\max}}\,\lambda_{\max} & \text{otherwise} \end{cases}$$

Task size scales with each BS's ingress traffic; below a small threshold ε no task is generated.

Edge resource sharing (GPS)

$$s_{(m,n)}(t) = \frac{f_n^{\mathrm{MEC}}\, D}{r_m\, B_n(t)}, \quad m \in \mathcal{B}_n(t)$$

Active queues at a server split its per-slot compute capacity equally.

Total system energy

$$E(t) = \sum_{m \in M} \left(E_m^{\mathrm{comp}}(t) + E_m^{\mathrm{tran}}(t) \right) + \sum_{n \in N} E_n^{\mathrm{MEC}}(t)$$

Local compute + transmission + MEC processing; compute energy grows with the cube of CPU frequency.

Long-term objective (Dec-POMDP)

$$\min_{\pi}\ \mathbb{E}_{\pi}\left[\sum_{t \in T_{\mathrm{ext}}} \gamma^{t-1} \left(\tilde{C}(t) + \mu E(t) \right) \right]$$

Minimize discounted delay-and-drop cost with a weighted energy penalty.

4 Intelligent Framework: ConvLSTM Prediction + MAPPO (CTDE)

① Spatiotemporal prediction — ConvLSTM encoder–decoder

Each OD traffic matrix is treated as a single-channel image, so the 2-D convolutions inside the gates capture spatial correlations between node pairs together with temporal dynamics. Trained offline (MSE, Adam, early stopping) and then frozen, the model forecasts next-slot traffic; the predicted task size $\hat{\lambda}_m(t+1)$ enters every BS observation as a look-ahead feature.

Input gate

$$\mathbf{i}_t = \sigma(\mathbf{W}_i * \mathbf{X}_t + \mathbf{U}_i * \mathbf{H}_{t-1} + \mathbf{b}_i)$$

Cell-state update

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c * \mathbf{X}_t + \mathbf{U}_c * \mathbf{H}_{t-1} + \mathbf{b}_c)$$

Hidden output

$$\mathbf{H}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

Prediction accuracy: **MAPE 8.92% · MAE 0.173 · RMSE 0.247 · R^2 = 0.877**

② Cooperative MARL — MAPPO (CTDE)

Offloading is a cooperative Dec-POMDP solved with MAPPO under centralized-training / decentralized-execution. A shared critic sees the global state (all observations plus an agent one-hot) in training, while each actor acts on its own observation at run time. An action mask skips empty slots; a delayed per-agent reward charges each penalty to the slot where the task finishes or drops; per-agent energy allocation conserves total energy.

Per-agent (delayed) reward

$$r_m(t) = -(\tilde{c}_m(t) + \mu E_m(t)), \qquad \sum_{m \in M} r_m(t) = R(t)$$

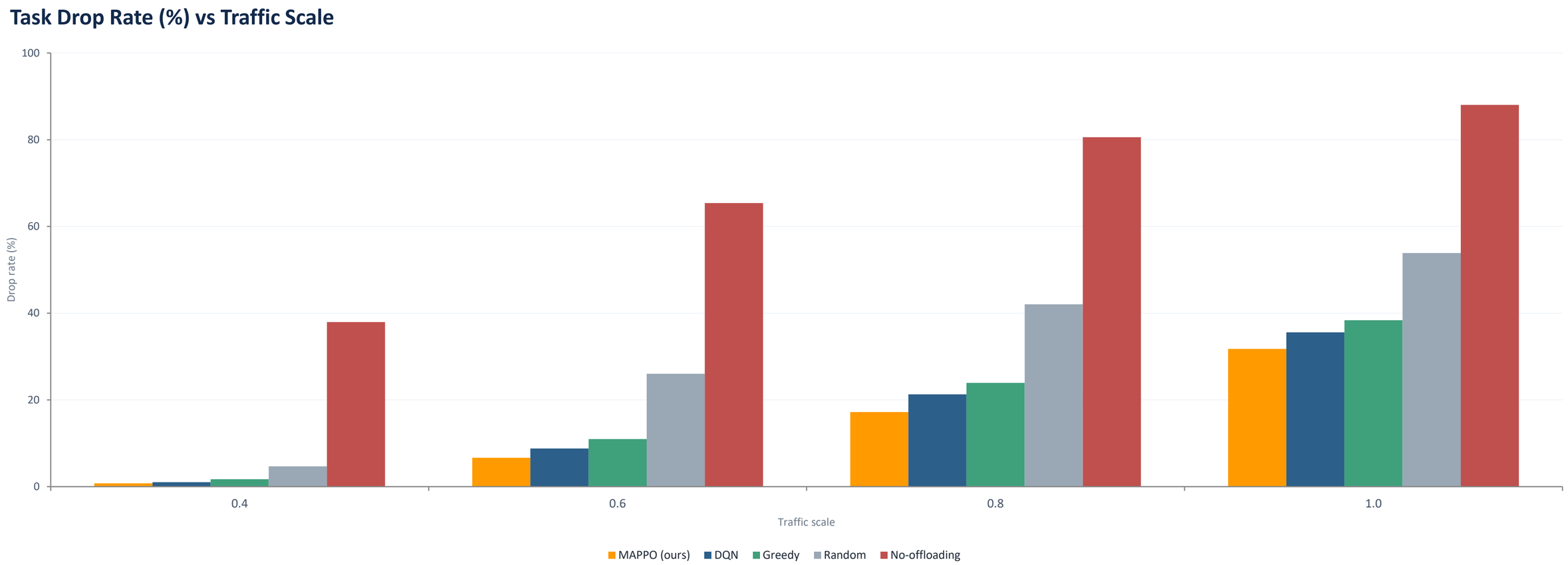
PPO-Clip surrogate objective

$$L^{\mathrm{CLIP}}(\theta_m) = \mathbb{E}_t\left[\min\left(\rho_{m,t}\hat{A}_{m,t},\ \mathrm{clip}(\rho_{m,t}, 1-\epsilon, 1+\epsilon)\hat{A}_{m,t}\right)\right]$$

Advantage — GAE

$$\hat{A}_{m,t} = \sum_{l=0}^{T_{\mathrm{end}}-t} (\gamma\lambda_{\mathrm{GAE}})^l \, \delta_{m,t+l}$$

5 Results: Consistent Gains Across Traffic Loads



KEY TAKEAWAYS

Proactive ConvLSTM features and CTDE coordination jointly cut delay, drop rate and energy against every baseline.

The drop-rate gap over single-agent DQN widens exactly when MEC contention is most severe — near saturation.

At load 1.0: 6.320 slots delay, 31.749% drop, 0.435 energy/task (−3.8 / −10.8 / −10.6% vs DQN).

FUTURE WORK

Replace the fixed wired BS–MEC link with a wireless channel and dynamic spectrum allocation.

Train MAPPO across MEC clusters of different operators without sharing raw traffic — privacy and scale.

Compress tasks at the semantic level before transmission to relieve fronthaul and MEC queueing pressure.

Conclusion

- An architectural framework that integrates artificial intelligence with edge and fog computing to enable autonomous operation in next-generation networks, including IMT-2030.
- Drawing from the Hierarchical Autonomous Edge Architecture, Edge Sandbox addresses a core challenge: delivering sophisticated self-management capabilities across highly heterogeneous fog/edge nodes, including those with severe resource constraints. By enabling local, sandboxed evolution and hierarchical orchestration across cloud, fog, and edge layers, this architecture substantially reduces the load on the core network while improving the efficiency of distributed learning.
- Processing data locally minimizes latency, conserves bandwidth, and reduces inter-network communication volume essential for scalable and responsive autonomous operations.
- The Edge Sandbox supports advanced applications such as telepresence, holography-type communications, autonomous vehicles, industrial IoT, and remote medical monitoring, enabling immediate and reliable responses to real-world events in autonomous network environments.

Thank You!