Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

PrivateAgent: Mobility-Aware AI Agents for Safe Private Networks

Onur Sahin

onur.sahin2@studbocconi.it

Bocconi University

Vanlin Sathya

Celona Inc

Lyutianyang Zhang

Chongqing University

Research Article

Keywords: Private networks, AI agents, edge intelligence, mobility-aware orchestration, network validation, trace-driven benchmarking

Posted Date: October 29th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10934448/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

PrivateAgent: Mobility-Aware AI Agents for Safe Private Networks

Onur Sahin[1], Vanlin Sathya[2], and Lyutianyang Zhang[3]

[1]Bocconi University, Milan, Italy
[2]Celona Inc., Cupertino, California, USA
[3]Chongqing University, Chongqing, China

Corresponding author: Onur Sahin, onur.sahin2@studbocconi.it
Vanlin Sathya: vanlin@celona.io Lyutianyang Zhang: zhanglyutianyang@cqu.edu.cn, czc@cqu.edu.cn

Abstract

Private cellular networks increasingly support video analytics, connected workers, autonomous mobile robots (AMRs), and other industrial applications whose mobility and control deadlines vary widely. A single artificial-intelligence (AI) agent cannot safely manage all of these loops because perception, planning, networking, and governance operate at different time scales and execution locations. This paper presents PrivateAgent, a mobility-aware framework that classifies private-network applications as stationary, low-mobility, or high-mobility and maps their functions across device, site-edge, and cloud tiers. Hard real-time perception and actuation remain local; near-real-time planning and network optimization run at the edge; long-horizon learning runs in the cloud. Network-changing actions are represented as typed artifacts with explicit evidence, scope, authority, monitoring, and rollback requirements before execution. We detail representative use cases including perimeter video, machine vision, connected workers, forklifts, AMRs, and drones, and specify suitable agent families, latency targets, failure modes, and benchmark dimensions. We also instantiate the action-admission boundary with SafeFlow-Real50, a 50-scenario benchmark derived from four anonymized private-network traces. In a blind evaluation, the model selected the correct act/clarify/reject mode in 50/50 scenarios, an acceptable action in 45/50, and the preferred action in 43/50. The deterministic gateway detected all 400 injected corruptions versus 150 for schema-only validation. These results show that mobility should determine the real-time agent boundary, while independent validation should determine whether generated network actions may advance toward execution.

Keywords: Private networks; AI agents; edge intelligence; mobility-aware orchestration; network validation; trace-driven benchmarking

1 Introduction

Private mobile networks are becoming an operational substrate for industrial automation rather than a narrow replacement for enterprise Wi-Fi. By the end of the third quarter of 2025, the Global mobile Suppliers Association had identified 1,907 organizations deploying private mobile networks with contract values above EUR 100,000 across 84 countries, with customer references growing at a compound annual rate of 34% since 2020 [1]. Manufacturing remains the largest deployment sector, followed by education and research, mining, and defense. A 2025 GlobalData study commissioned by Nokia surveyed 115 industrial enterprises and reported that 87% of private-wireless and on-premise-edge adopters obtained a return on investment within one year and 68% did so within six months [2]. On the capital side, 81% reported lower setup costs than alternative connectivity approaches and 55% reported a reduction of at least 11%. On the operating side, 86% reported lower ongoing costs and 60% reported an OPEX reduction of at least 11%; one respondent attributed approximately US$5 million in annual savings to fewer truck rolls. Seventy percent were already running AI-enabled applications on the combined edge and connectivity platform, and 41% had expanded the network to use cases beyond the original

deployment. The same report forecasts that the global private-wireless market will approach US$8 billion by 2027. These figures should be interpreted as industry-reported evidence rather than independent causal estimates, but they show why enterprises evaluate private networks through operational outcomes: fewer truck rolls, less cabling, lower downtime, safer workers, better material flow, and scalable automation.

The application mix is broad. Fixed perimeter cameras stream video continuously but rarely change attachment point. A connected worker moves at walking speed while using voice, video, augmented reality, and location services. An automated guided vehicle (AGV) or autonomous mobile robot (AMR) moves through handover regions while simultaneously localizing, avoiding obstacles, receiving fleet tasks, speaking to workers, and transmitting 360-degree video. These workloads cannot be managed by a single undifferentiated "AI agent." The stationary camera can tolerate second-scale policy adjustments if video remains buffered, while the AMR collision-avoidance loop cannot wait for a remote large language model (LLM). A useful architecture must therefore answer four linked questions:

1. **What** intelligence is required: perception, anomaly detection, dialogue, planning, mobility prediction, network optimization, or governance?

2. **When** should each agent run: periodically, on an event, before a handover, after an alarm, or continuously in a hard real-time loop?

3. **Where** should it run: on the endpoint, at the site gateway or edge, or in the cloud?

4. **How** should its proposed network action be checked before execution?

AI-native networking architectures increasingly place learning and reasoning inside configuration, orchestration, and recovery [3–6]. LLM-based systems can interpret operator intent, correlate heterogeneous evidence, diagnose faults, and generate candidate policies [7–10]. However, two engineering errors are often conflated. The first is choosing the wrong agent or placing it in the wrong execution tier; this produces missed deadlines, unnecessary backhaul, privacy exposure, or unstable control. The second is admitting an unsafe action from an otherwise capable agent; this produces invalid paths, excessive scope, tenant leakage, security-waypoint bypass, or an unrecoverable configuration. A private network that supports robots and safety-sensitive applications must solve both.

This paper develops a unified use-case and control framework rather than treating connectivity, AI placement, and validation as separate topics. It extends prior surveys of private-network evolution, use cases, spectrum, and architecture [11] by organizing applications according to mobility and control deadlines. It also extends LLM-network-management work by introducing a common device-edge-cloud agent hierarchy and by grounding the safety layer in a reusable trace-driven benchmark.

The contributions are:

- **Mobility-aware use-case taxonomy.** We divide private-network applications into stationary, low-mobility, and high-mobility classes and provide detailed operational profiles for cameras, connected workers, mobile scanners, forklifts, AGVs, AMRs, and drones.
- **Agent and placement map.** For each class, we identify suitable perception, prediction, language, planning, and network-management agents; define when they should run; propose device, site-edge, or cloud placement; and give design-time latency budgets, advantages, disadvantages, and failure impacts.
- **Common agentic framework.** PrivateAgent separates the safety-critical local loop from edge coordination and cloud learning, while exposing a typed network-action contract with evidence, authority, monitoring, and rollback.
- **Benchmarking methodology.** We specify category-specific benchmarks for deadline satisfaction, mobility continuity, perception quality, network action safety, resource cost, and recovery, and release SafeFlow-Real50 as an initial trace-grounded benchmark for the network-action admission layer.
- **Quantitative validation.** A blind action-generation experiment obtains 100% control-mode accuracy, 90% acceptable-action accuracy, and 86% preferred-action accuracy. Deterministic validation detects 400/400 injected corruptions compared with 150/400 for schema checking alone.

The remainder of the paper is organized as follows. Section 2 reviews private-network, AI-native, agentic-management, validation, and trace-benchmarking work. Section 3 develops the three mobility

classes and their use cases. Section 4 presents the common architecture and placement logic. Section 5 describes SafeFlow-Real50 and the evaluation method. Section 6 reports results and design implications. Section 7 concludes with future work.

2 Background and Related Work

2.1 Private networks and industrial use cases

Private networks provide enterprise-controlled coverage, security, quality of service, local data handling, and spectrum options that are difficult to guarantee with best-effort Wi-Fi or shared public mobile access. The evolution from private LTE toward private 5G expands the feasible use cases from basic sensing and communications to high-rate video, connected workers, robotics, industrial IoT, and edge intelligence [11]. The value does not arise from connectivity alone. It comes from coupling radio coverage with local compute, device identity, policy, and operational applications. In industrial environments, automation traffic is also heterogeneous: camera streams are bandwidth-heavy and mostly stationary; handheld workflows are interactive and mobility-sensitive; and robots combine periodic control, bursty video, map updates, and fleet coordination.

The prior private-network literature explains architecture, spectrum, deployment models, and vertical applications in depth [11]. What remains underdeveloped is a systematic mapping from each use case to the AI agent type, execution tier, invocation timing, network-management action, and validation method. This mapping becomes critical when the network no longer carries only data but also carries decisions that alter physical behavior.

2.2 AI-native and LLM-assisted management

AI-native network proposals embed intelligence in the control and management plane. Hossain et al. study AI-native configuration for the 6G core, while REASON organizes AI-native control across multi-access networks [3, 4]. NetGPT and the AI-native interconnect framework couple generative services, edge intelligence, and future-network orchestration [5, 6]. These architectures motivate model-driven control but generally leave the application-specific real-time boundary open.

LLM-based operations research spans diagnosis, policy generation, incident handling, and recovery. Hamadanian et al. position AI as an iterative assistant for network incident management rather than an unconstrained replacement for operators [7]. LORADOCTOR combines diagnosis with adaptive LoRaWAN policy optimization [8]. NetPrompt translates intent into programmable policies [9], and Huang et al. use LLMs for autonomous recovery in AI-native networks [10]. Surveys by Liu et al. and Hong et al. identify grounding, trust, validation, and feedback as central requirements for LLM-based network operations [12, 13]. NeTestLLM shows how executable protocol tests and runtime feedback can constrain agent-generated artifacts [14].

These studies establish that LLMs can assist management, but they do not imply that an LLM belongs in every control loop. A language model is well suited to intent parsing, multi-step planning, explanation, and retrieval over procedures. It is poorly suited to deterministic braking, millisecond obstacle avoidance, or direct radio-actuator control. The appropriate role therefore depends on mobility, deadline, model size, state freshness, and consequence of failure.

2.3 Validation, policy enforcement, and agent security

Hybrid validation pipelines combine generative reasoning with deterministic checks. Raptis et al. validate intent syntax, semantics, and policy compliance before network automation [15]. Petrovic et al. combine LLM-driven management with formally checkable design models for IoT systems [16]. Policy checkers for industrial SDN and verifiable IoT enforcement show that network-wide invariants can be tested independently of the component that generated the configuration [17, 18]. AgentArmor applies program analysis to runtime traces to detect prompt-injection-induced trust violations [19].

The shared principle is separation of duties: the generator proposes, while an independent component verifies authority, scope, and invariants. Our framework applies that principle to private-network agents and adds mobility-aware placement. The verifier is not only a security filter; it is also the boundary that prevents a slow or semantically capable agent from entering a control domain whose deadline or authority it cannot satisfy.

2.4 Trace-driven and infrastructure benchmarks

Trace-driven evaluation preserves realistic workload variation while allowing repeatable experiments. Classical work shows how trace replay improves workload modeling and simulation realism [20, 21]. FaaS-sim applies the same principle to serverless edge computing [22]. Recent LLM-infrastructure work uses topology-aware testers, traces, and resilience analysis to evaluate distributed training and inference [23–26]. These benchmarks focus on LLM infrastructure rather than LLM-generated network actions, but they demonstrate the importance of topology, timing, and reproducibility.

2.5 Gap addressed by this paper

Existing work covers private-network use cases, AI-native architecture, LLM-assisted management, deterministic policy checking, and trace-based benchmarking as largely separate lines. No prior framework in this body of work jointly:

1. categorizes private-network applications by stationary, low-mobility, and high-mobility behavior;
2. maps each category to the agent type, trigger, execution location, runtime budget, and failure impact;
3. defines a common device-edge-cloud orchestration model for multimodal agents;
4. treats generated network actions as typed, independently verifiable artifacts; and
5. supplies a reusable trace-driven benchmark and hidden oracle for evaluating the admission boundary.

This combined gap is the focus of PrivateAgent and SafeFlow-Real50.

3 Mobility-Aware Private-Network Use Cases

3.1 Taxonomy dimensions

Mobility alone does not determine system design, but it is an effective first-order classifier because it changes channel coherence, handover frequency, location uncertainty, and the time available to correct a bad decision. We characterize each use case with five dimensions:

1. **Mobility**: stationary, walking or vehicle-low-speed, or high-mobility robotic motion;
2. **Control deadline**: hard real time, soft real time, near real time, or non-real time;
3. **Modality**: scalar telemetry, audio, image/video, map, language, or multimodal fusion;
4. **Action criticality**: informational, operational, safety-relevant, or physical-actuation critical;
5. **Network dependence**: whether the application can continue locally during an outage and for how long.

Table 1: Mobility-aware private-network use-case taxonomy and agent requirements. Runtime values are design targets for the indicated decision loop, not measurements from the field traces

Class	Representative use cases	Dominant traffic	Required agents	Preferred placement	Decision budget	Failure impact
Stationary	Perimeter camera, fixed quality inspection, machine-health sensing	Continuous video; periodic telemetry	Vision inference, event correlation, anomaly detection, bandwidth/QoS agent	Camera or edge for inference; site edge for correlation; cloud for training	100 ms–5 s for alarms; minutes for retraining	Missed intrusion, quality escape, false maintenance alarm, wasted uplink
Low mobility	Connected worker, body-worn camera, AR maintenance, handheld scanner, forklift	Interactive audio/video; bursty uploads; location updates	Speech/vision agent, context retrieval, handover predictor, session/QoS agent	Device for capture and safety cues; edge for multimodal fusion and retrieval	50–500 ms interaction; 0.5–2 s planning	Stale instruction, interrupted workflow, privacy leakage, unsafe guidance
High mobility	AGV, AMR, autonomous forklift, drone	Control telemetry; maps; multi-camera; fleet messages	Local perception/control, trajectory planner, fleet coordinator, mobility/network agent, verifier	Onboard for safety loop; edge for fleet and network coordination; cloud for learning	5–50 ms safety; 50–200 ms path update; seconds for task planning	Collision, route deadlock, production stop, fleet cascade, loss of control

3.2 Stationary use cases

3.2.1 Perimeter camera monitoring

Perimeter cameras are common at factories, warehouses, ports, campuses, and critical infrastructure. Today they are used for intrusion detection, restricted-zone monitoring, license-plate or object recognition, safety-rule enforcement, and forensic recording. A private network is useful when cameras are widely distributed, cabling is expensive, Wi-Fi coverage is inconsistent, or video must remain within the enterprise domain. The application may generate a sustained high-rate stream, but the control problem is comparatively stable because the device location and expected radio environment change slowly.

A practical agent stack has four levels. A compact vision model on the camera or nearby edge filters frames and detects candidate events. A site-level correlation agent combines events from multiple cameras, access-control systems, and location sensors. A network agent changes bitrate, uplink priority, or path only when congestion or degradation threatens an active alarm. An LLM or vision-language model can summarize an incident, query procedures, or explain evidence, but it should not directly disable a camera or open a security zone. Typical invocation is event-driven: the perception agent runs continuously, the correlation agent runs on detections, and the network agent runs on sustained loss, queue growth, or stream-quality violation.

The main placement trade-off is bandwidth versus model capacity. On-camera inference reduces uplink load and preserves privacy but is constrained by power and memory. Edge inference supports larger models and cross-camera fusion at low latency. Cloud processing enables expensive analytics and retraining but adds transport dependency and data-governance exposure. A useful benchmark must therefore measure not only detection quality but also event-to-alarm latency, video loss under congestion, false action rate, privacy boundary, and behavior during backhaul failure. Figure 1 illustrates the intended control boundary: perception triggers the request, but the private-network controller accepts only a temporary, scoped action whose bandwidth and privacy consequences have been checked.

3.2.2 Fixed machine vision and quality inspection

Machine-vision stations inspect welds, labels, packaging, surface defects, dimensions, or assembly completeness. Unlike perimeter video, the decision is tied to a production cycle. The inference path may require a deterministic deadline so that a defective item can be diverted before it leaves the station. The endpoint is stationary, but the object stream is fast and repetitive.

The recommended architecture places the primary inference model on the industrial PC or local edge, with a deterministic rule or programmable logic controller retaining the final reject authority. A second

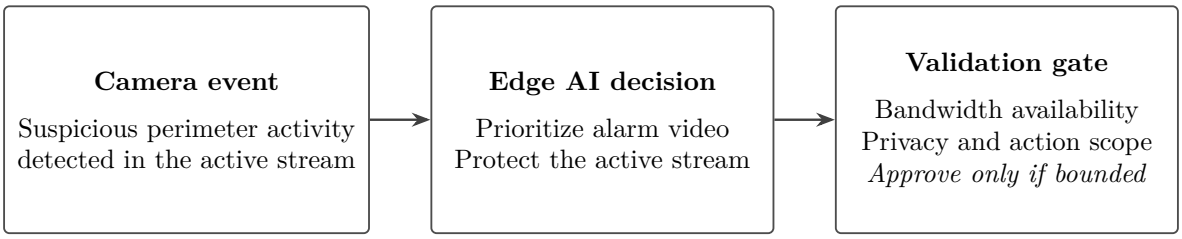

Figure 1: Perimeter-video workflow in which an event-driven edge agent requests temporary network protection while an independent gate checks bandwidth, privacy, and action scope before execution

agent detects data drift, lighting changes, or camera degradation. A network agent reserves or protects the inspection flow only when local inference depends on remote acceleration. An LLM is useful for explaining defect patterns, generating maintenance tickets, or retrieving inspection procedures; it is not the sole classifier for a safety- or quality-critical reject decision. Benchmarks should combine defect-detection accuracy, cycle-time deadline misses, data-drift sensitivity, network interruption tolerance, and the rate of unjustified network actions.

Figure 2 shows the corresponding network-management loop. When defect-analysis latency approaches its cycle budget, the edge agent may either protect the inspection stream or move inference closer to the line. The validation gate must verify capacity, placement feasibility, rollback, and preservation of other critical production traffic before the change is applied.

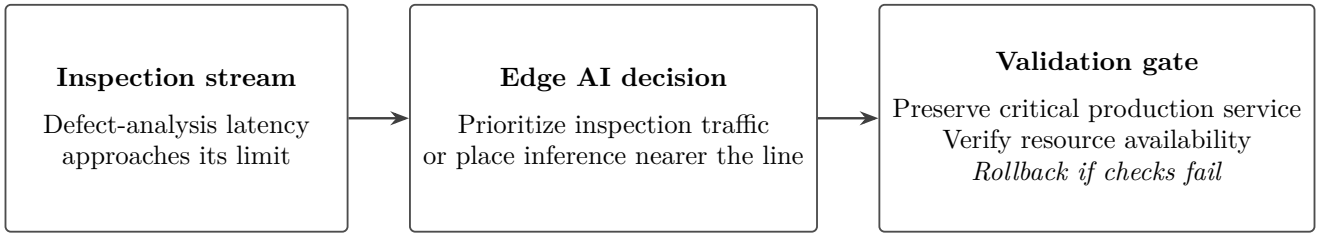

Figure 2: Machine-vision workflow in which the edge agent responds to a tightening inference deadline with a bounded QoS or placement request that must preserve other critical production traffic

3.2.3 Machine-health and environmental monitoring

Fixed sensors measure vibration, temperature, pressure, current, gas, or acoustic signatures. Traffic volume is low, but scale can be large. Time-series models, change-point detectors, and causal diagnosis agents are usually better than LLMs for the first-stage anomaly loop. An LLM can aggregate alarms, retrieve historical incidents, and create a maintenance plan. Placement can be split: simple detection on the sensor or gateway, correlation at the site edge, and fleet-level learning in the cloud. Wrong placement creates either unnecessary uplink load or delayed detection. Wrong reasoning can create false shutdowns, missed maintenance, or repeated truck rolls. Benchmarks should measure time-to-detect, false-alarm rate, missing-data robustness, maintenance-cost proxy, and whether the agent can distinguish a network fault from a physical-machine fault.

3.3 Low-mobility use cases

3.3.1 Connected worker and augmented maintenance

Connected workers use rugged handhelds, headsets, or augmented-reality devices for instructions, remote expert support, checklists, and digital work orders. The user typically moves at walking speed, which makes session continuity and local context more important than hard millisecond control. A multimodal agent may combine speech recognition, image understanding, retrieval over manuals, location, and network state.

A safe decomposition is: keyword spotting and capture control on the device; speech recognition, document retrieval, and vision-language reasoning at the site edge; long-horizon knowledge-base maintenance in the cloud. The device must continue to present a stable procedure when connectivity degrades. The edge agent may prefetch the next task steps before the worker enters a weak-coverage zone. A network agent can prioritize the active session or steer it between private cellular and Wi-Fi, but only with evidence that the target access path is available. A language agent should return a clarification when the task, equipment identifier, or authorization is missing.

The most harmful wrong-agent behavior is not always an outage. A confident but stale instruction can be worse than no instruction. A useful benchmark therefore combines interaction latency, speech and image quality, handover interruption, retrieval freshness, authorization correctness, and the percentage of sessions that safely degrade to offline mode. Figure 3 shows how the edge agent can choose between local response and cloud retrieval while keeping authorization, privacy, and temporary low-latency support in a separate admission path.

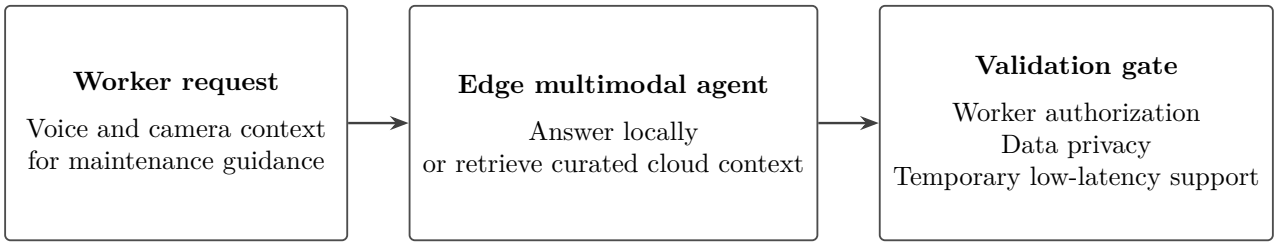

Figure 3: Connected-worker workflow in which a site-edge multimodal agent chooses local response or curated cloud retrieval while authorization and privacy checks bound any temporary low-latency network support

3.3.2 Body-worn cameras and worker safety

Body-worn cameras combine mobility, continuous video, privacy, and incident evidence. A local model can detect a fall, missing protective equipment, or entry into a restricted zone. The edge correlates the alert with geofencing, badge identity, and nearby machinery. The network agent may prioritize the alarm clip and reduce noncritical video quality. Cloud storage and training are appropriate only after policy-based redaction and retention checks. A wrong agent can expose personal data, suppress an alarm, or flood the network with full-resolution video. Benchmarking must include privacy-preserving accuracy, alarm latency, geofence correctness, uplink adaptation, and audit completeness.

3.3.3 Handheld scanning, asset tracking, and forklifts

Warehouse workers and forklifts generate barcode, inventory, telemetry, and occasional video traffic. Mobility is low to moderate, but radio shadowing between racks can be severe. The best agent is often predictive rather than generative: it estimates the next coverage zone, prefetches work, and schedules a handover or access switch. An LLM can interpret a workflow exception, but the network action should be a typed request with a declared traffic class and duration. Benchmarks should replay routes, rack-induced coverage changes, and task deadlines, then measure session completion, handover loss, duplicate transactions, and unsafe access expansion.

3.4 High-mobility use cases

3.4.1 AGVs and AMRs

AGVs follow constrained paths, whereas AMRs build or update maps and plan routes dynamically. Both can combine lidar, inertial sensing, wheel odometry, cameras, microphones, speakers, and private-network connectivity. A modern AMR may receive a spoken task, identify a pallet through vision, navigate to a

destination, coordinate with doors or elevators, and report video to a remote operator. No single agent should control all of these functions at one time scale.

The onboard stack must own the safety envelope: obstacle detection, emergency stop, local localization, short-horizon trajectory control, and a fallback behavior during network loss. These tasks need deterministic or compact learned models and design budgets commonly in the 5–50 ms range. The site edge can run map fusion, fleet coordination, speech recognition, vision-language task grounding, and medium-horizon route planning in the 50–500 ms range. The cloud can train models, analyze fleet-wide trends, and optimize schedules over seconds to hours. An LLM may translate "move pallet 17 to line 3 and report damage" into a structured task plan, but each physical action must be bound to capabilities, location, authorization, and a monitored execution state.

The network agent should run proactively. It can predict that an AMR is approaching a weak-coverage or handover region, reserve resources, move an inference session, or pre-stage a map. It must not reroute management or safety traffic based only on a strong alternative signal; target-path availability, security waypoint, tenant isolation, and rollback must be verified. A wrong placement or wrong agent can create collision risk, route oscillation, fleet deadlock, video loss, or a production stop. Figure 4 makes the placement decision explicit: the robot can re-plan locally, offload perception, or request handover, but its local safe-stop capability remains the non-negotiable fallback.

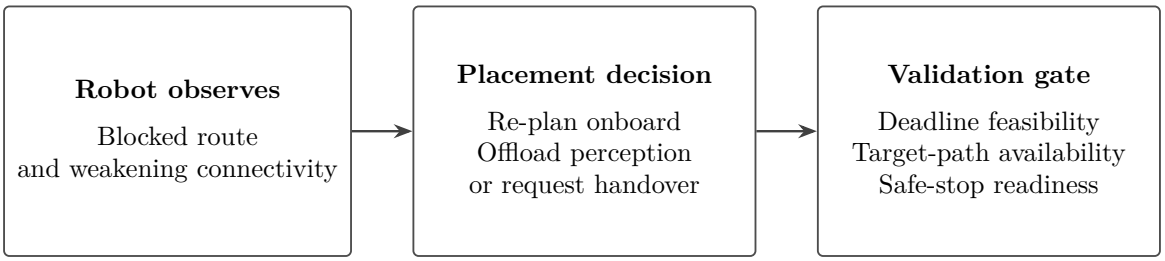

Figure 4: AGV/AMR workflow in which mobility and connectivity evidence drive a local-planning, offload, or handover choice, while deadline, path-availability, and safe-stop checks remain mandatory before a network action is admitted

3.4.2 Autonomous forklifts and industrial vehicles

Autonomous forklifts combine high mass, occluded environments, and mixed human traffic. Their safety loop must remain local. The edge can coordinate intersections, assign tasks, and fuse infrastructure cameras with onboard perception. A private network can carry cooperative awareness and high-rate video, but a remote model must never be the sole braking authority. Benchmarks should include packet delay and loss bursts, handover timing, occlusion, route conflicts, emergency-stop success, and recovery from edge disconnection.

3.4.3 Drones and aerial inspection

Drones inspect roofs, tanks, pipelines, yards, or hazardous zones. Mobility is higher and three-dimensional, and the radio path can change quickly. Onboard stabilization and collision avoidance are mandatory. The edge can run heavy vision, mission replanning, and geofence checks. The cloud is appropriate for historical comparison and model training. Network actions must consider command-link protection separately from payload video. A bandwidth optimization that degrades the command channel is unacceptable even if aggregate throughput improves. Benchmarks should measure command continuity, geofence compliance, video utility, handover recovery, and safe return-to-home behavior.

Figure 5 illustrates the required separation between mission intelligence and network authority. Under a degrading link, the agent can analyze onboard, send selected frames to the edge, or reduce bitrate; however, battery level, command-link quality, flight safety, and return-to-home readiness must be checked before adaptive streaming or path changes are admitted.

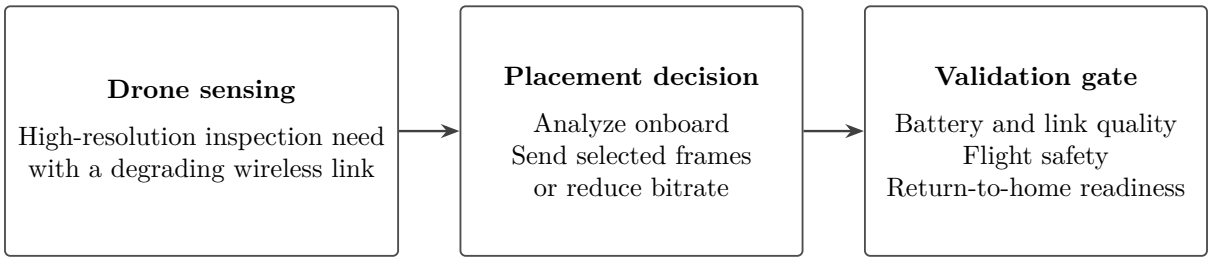

Figure 5: Drone-inspection workflow in which the agent adapts analysis placement or streaming intensity only after battery, link-quality, flight-safety, and return-to-home checks

3.5 Algorithm, placement, and execution trade-offs

Table 2 summarizes the main algorithm families. "Runtime" denotes an engineering target for one decision cycle on the chosen tier; it is not a universal model benchmark. The correct choice is usually hierarchical. Compact perception and control models meet hard deadlines onboard. Specialized prediction, retrieval, and coordination agents run at the site edge. LLMs and vision-language models provide semantic planning when a subsecond or second-scale deadline is acceptable. Cloud services train models and optimize over long horizons.

3.6 Bottlenecks caused by the wrong agent

The most important bottlenecks are architectural rather than purely model-accuracy problems:

- **Deadline mismatch:** a remote LLM is placed in a loop that requires deterministic millisecond response.
- **State staleness:** an agent reasons over delayed radio, map, queue, or policy state and selects a path that no longer exists.
- **Capability mismatch:** a general language model is used where a time-series detector, planner, or deterministic controller is more appropriate.
- **Resource contention:** several agents compete for the same accelerator, causing unpredictable inference delay.
- **Mobility blindness:** the agent optimizes current signal without considering trajectory, handover, or session migration.
- **Authority inflation:** a diagnostic or advisory agent is granted execution rights.
- **Missing validation:** a syntactically valid action violates scope, policy, or rollback requirements.

For a stationary camera, these errors waste bandwidth or miss an alarm. For a connected worker, they create stale or unsafe guidance. For an AMR, they can propagate into physical motion and halt an entire fleet. The framework in the next section places explicit boundaries around these risks.

Table 2: Agent algorithm families for private-network applications. Runtime ranges are design targets that depend on hardware, model size, and batching

Agent family	Suitable tasks	Typical tier	Target cycle	Advantages	Disadvantages / safeguards
Deterministic control / state machine	Emergency stop, actuator interlock, fail-safe mode	Device	1–20 ms	Predictable, auditable, certifiable	Limited adaptation; must remain final safety authority
Compact vision / audio model	Object detection, fall detection, keyword spotting	Device or edge	5–100 ms	Low latency, privacy, reduced uplink	Narrow task; drift monitoring and signed updates required
Time-series anomaly detector	Machine health, radio anomaly, queue growth	Gateway or edge	10 ms–1 s	Efficient, strong for structured telemetry	Can confuse network and physical faults; needs causal context
Mobility / handover predictor	Coverage transition, route-aware prefetch, access selection	Edge	20–200 ms	Proactive, uses trajectory and radio history	Sensitive to stale maps and distribution shift
Model-predictive / graph planner	Robot route, fleet conflict, compute placement	Device or edge	10–500 ms	Constraint-aware, interpretable objectives	Model mismatch and computational scaling
Retrieval-augmented language agent	Procedures, incident triage, task decomposition	Edge or cloud	0.2–5 s	Handles unstructured intent and documentation	Hallucination, stale retrieval, non-determinism; no direct actuator authority
Vision-language / multimodal agent	Scene explanation, task grounding, remote assistance	Edge or cloud	0.3–10 s	Integrates image, speech, and language	Compute-heavy; privacy and deadline concerns
Multi-agent coordinator	Cross-domain planning, fleet/network joint decisions	Edge or cloud	0.5–30 s	Specialization and parallelism	Conflict, message overhead, cascading errors; requires arbitration

4 A Common Mobility-Aware Agentic Framework

4.1 Three execution tiers and nine agent roles

PrivateAgent uses three execution tiers: device, site edge, and cloud. The device tier owns hard real-time perception, actuation, and safe fallback. The site edge owns cross-device context, multimodal reasoning, fleet coordination, mobility prediction, and most network-control decisions. The cloud owns model training, long-horizon analytics, cross-site optimization, and governance updates. A task may migrate between tiers only when its deadline, privacy, compute, and continuity constraints remain satisfied.

The framework separates nine roles:

1. **Perception agent**: converts sensor data into structured state.
2. **Context agent**: fuses topology, radio, compute, location, service, and policy state.
3. **Intent agent**: translates operator or application goals into a typed objective.
4. **Planner agent**: proposes a task or network plan.
5. **Mobility agent**: predicts attachment, handover, route, and session-continuity risk.
6. **Network agent**: proposes QoS, path, access, rate, isolation, or placement actions.
7. **Verifier**: checks evidence, ontology, policy, scope, deadline, authority, and rollback.
8. **Execution and monitoring agent**: applies an approved action and tests the expected effect.
9. **Reflection agent**: records outcomes and proposes offline improvements without changing production policy directly.

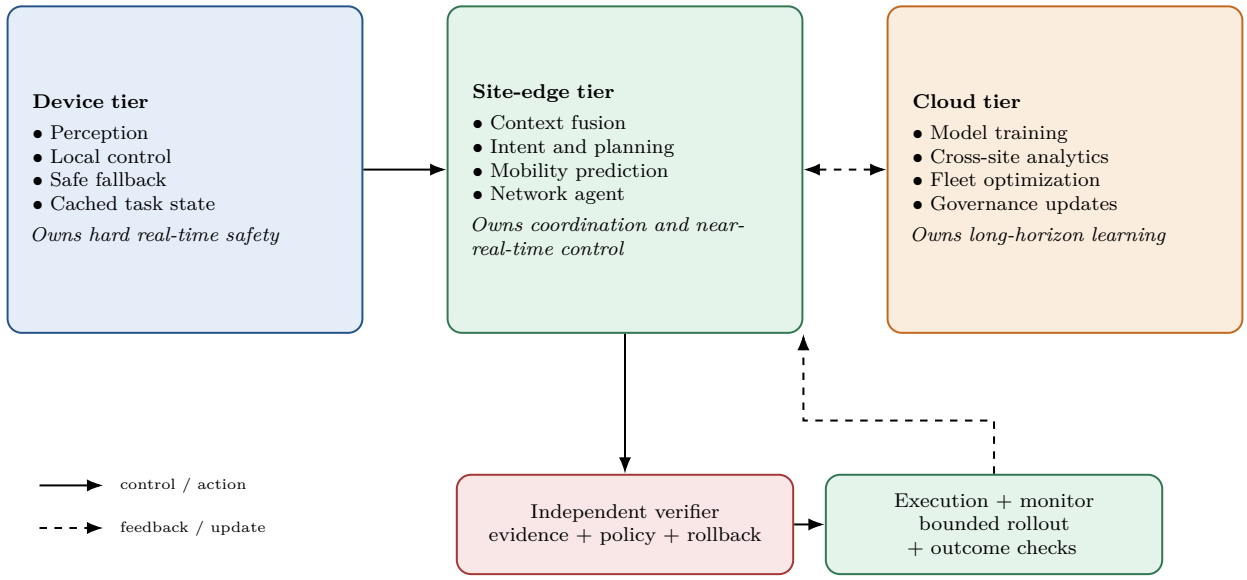

Figure 6: PrivateAgent separates hard real-time endpoint autonomy, site-level agent coordination, and cloud learning. The verifier is outside the generating agent chain and controls advancement toward execution

4.2 What, when, and where selection

For an agent task i and candidate execution tier $p \in \{\text{device}, \text{edge}, \text{cloud}\}$, placement can be represented as a constrained objective:

$$\min_{p} \quad w_L L_{i,p} + w_E E_{i,p} + w_B B_{i,p} + w_P P_{i,p} + w_C C_{i,p} \tag{1}$$

$$\text{s.t.} \quad L_{i,p} \le D_i, \quad A_{i,p} \ge A_i^{\min}, \tag{2}$$

$$R_{i,p} \ge R_i^{\min}, \quad p \in \mathcal{P}_i. \tag{3}$$

Here L is end-to-end decision latency, E endpoint energy, B network bandwidth, P privacy exposure, C compute or service cost, D_i the task deadline, A model quality, R availability, and $\mathcal{P}_i$ the tiers authorized for the task. Hard real-time control restricts $\mathcal{P}_i$ to the device. Privacy-sensitive video may restrict it to the device and site edge. Long-horizon training usually selects the cloud.

Invocation policy is equally important. Continuous agents are reserved for perception and safety. Periodic agents estimate health or capacity. Event-driven agents respond to alarms and intent. Predictive agents run before a route transition or capacity threshold. Reflective agents run after an incident and cannot alter production state without a separate approval cycle.

4.3 Typed actions and independent validation

The network agent emits a typed artifact rather than raw controller code. The current action vocabulary contains reroute, access switch, QoS priority, rate limit, block, scoped allow, endpoint isolation, configuration restore, monitor, clarification, rejection, and human approval. Each action declares its target, scope, duration, evidence, expected effect, risk, and rollback.

The verifier performs seven stages:

1. catalog and schema binding;
2. ontology membership for paths, endpoints, traffic classes, and policies;
3. evidence-provenance and freshness checks;
4. scope, duration, numerical, and resource bounds;

Table 3: Minimum benchmark dimensions for mobility-aware private-network agents

Dimension	Stationary	Low mobility	High mobility
Environment	Fixed radio and scene changes; congestion; camera or sensor loss	Walking routes; rack or wall shadowing; access transitions	Time-varying route, handover, occlusion, multi-robot contention
AI metrics	Detection quality, false alarms, drift, event-to-alarm latency	ASR/VLM quality, retrieval freshness, interaction latency	Perception deadline, localization, planning latency, emergency-stop success
Network metrics	Stream loss, throughput, queue delay, QoS preservation	Session interruption, handover loss, access-selection stability	Control-channel continuity, jitter, route-aware migration, packet bursts
Safety metrics	Missed alarm, policy breach, privacy leak	Unsafe instruction, authorization error, stale context	Collision proxy, deadlock, unsafe motion, loss of command
Governance	Evidence, scope, retention, rollback	Identity, location, purpose, human approval	Capability binding, route authorization, local fallback, rollback
Cost and energy	Camera compute, uplink, storage	Device battery, edge inference, backhaul	Onboard energy, accelerator contention, fleet compute

5. authority and risk-class consistency;

6. topology and policy invariants such as isolation, loop freedom, management reachability, and security-waypoint traversal;

7. rollback completeness and runtime-monitor definition.

A low-risk monitoring action may pass local checks immediately. A high-risk access or isolation change may require simulation, canary execution, and human approval. A safety-critical robot action is never compiled directly from an LLM response; it is translated into a bounded task for the local controller.

4.4 Mobility-aware scheduling and graceful degradation

The framework uses a two-dimensional priority: physical deadline and operational risk. A local collision-avoidance task has the highest deadline priority but does not require an LLM. A tenant-access change has a slower deadline but high governance risk. The scheduler therefore reserves deterministic compute for device safety, bounded edge capacity for mobility and network agents, and opportunistic capacity for language and reflection agents.

Graceful degradation is explicit. If the edge is unavailable, a stationary camera continues local detection and buffering; a connected worker uses cached procedures and blocks unverified updates; an AMR slows or stops according to its local safety policy. Loss of the cloud should not stop the site. Loss of the site edge should not remove the device safety envelope.

4.5 Benchmark requirements by mobility class

A credible evaluation must measure more than model accuracy. Stationary benchmarks need sustained-stream behavior, event latency, bitrate adaptation, and privacy. Low-mobility benchmarks need handover interruption, session continuity, retrieval freshness, and user interaction. High-mobility benchmarks need deadline misses, trajectory error, collision or emergency-stop behavior, fleet deadlock, handover recovery, and safe fallback. All classes need network-action correctness, policy violations, rollback success, cost, energy, and audit completeness.

5 Trace-Driven Benchmark and Evaluation Method

5.1 Benchmark purpose and public release

SafeFlow-Real50 is the initial benchmark for the framework's network-action admission boundary. It does not claim to reproduce robot motion or camera perception. Instead, it evaluates whether an LLM selects an appropriate bounded management action from trace-grounded telemetry and whether deterministic validation catches unsafe artifacts before execution. To support independent replication and extension,

Table 4: Trace sources used to construct SafeFlow-Real50

Deployment	Raw rows	Duplicates	Clean events	Scenarios	Device / access	Available evidence
Health	1,024	512	512	12	Pixel 5a; private LTE + Wi-Fi	Serving and neighbor cells, visible/connected Wi-Fi, 137 RTT and 31 throughput samples
Headquarters	3,221	1,556	1,665	12	Pixel 6; private LTE + Wi-Fi	Dense LTE and Wi-Fi visibility; no active RTT or throughput tests
Supercenter	6,010	0	1,468	14	Galaxy XCover Pro; public/private LTE	Spatially matched AT&T and CBRS observations; multi-cell events; no active performance tests
Distribution center	899	253	646	12	Pixel 5a; private LTE + visible Wi-Fi	Serving/neighbor cells and visible Wi-Fi; no connected-Wi-Fi or active performance fields
Total	11,154	2,321	4,291	50	Four anonymized environments	Real radio evidence with explicit missing fields

the benchmark scenarios, hidden oracle, generated actions, deterministic evaluator, mutation engine, and paper artifacts are available at `https://github.com/OnurSahh/Trace2Action_GitHub_Repo`.

The benchmark contains exactly 50 blind scenarios derived from four anonymized deployments: a health-related private network, a headquarters or office campus, a large-format retail supercenter with public and private cellular measurements, and a distribution-center environment. The raw traces contain 11,154 rows. After duplicate removal and event grouping, 4,291 normalized observations remain. Table 4 summarizes the sources.

5.2 Scenario construction and hidden oracle

The public dataset includes telemetry windows, deployment context, a detected condition, an operator intent, ten safety policies, a fixed network ontology, and twelve permitted action families. It excludes expected decisions and correct answers. The hidden oracle is generated separately using deterministic rules over intent specificity, telemetry completeness, policy conflict, and action constraints.

The scenarios are balanced across five categories: cellular coverage or quality degradation, multi-access or Wi-Fi steering, public/private access selection, cell reselection or restoration, and ambiguous or policy-sensitive requests. The hidden decision distribution is 25 actionable, 15 unsafe-reject, and 10 clarification-required. Actionable cases contain a set of acceptable actions and one preferred action because conservative alternatives may coexist. For example, monitoring can be acceptable when an access switch is plausible but performance evidence is incomplete.

5.3 Action-generation and validation protocol

A single stateless model call receives the public file and returns one typed action per scenario. The hidden oracle is joined only afterward. The evaluator reports:

$$A_{\text{mode}} = \frac{1}{n}\sum_i \mathbb{1}[\hat{d}_i = d_i], \tag{4}$$

$$A_{\text{acceptable}} = \frac{1}{n}\sum_i \mathbb{1}[\hat{a}_i \in \mathcal{A}_i], \tag{5}$$

$$A_{\text{preferred}} = \frac{1}{n}\sum_i \mathbb{1}[\hat{a}_i = a_i^*]. \tag{6}$$

Mode accuracy tests whether the model acts, clarifies, or rejects. Acceptable-action accuracy tests whether the action family is defensible. Preferred-action accuracy is the strictest semantic measure.

The gateway checks catalog binding, required fields, ontology values, evidence equality, finite bounds, rollback objects, and risk consistency. To test coverage, each of the 50 valid artifacts is mutated once by eight deterministic operators: unknown action, action-name mismatch, missing field, fabricated evidence,

ontology violation, invalid bound, rollback or semantic corruption, and risk mismatch. This creates 400 known-bad artifacts. A schema-only baseline checks only action identity and required fields.

Large language model use. The blind candidate-action stage used GPT-5.6 Thinking in one stateless batch call to generate exactly one structured action for each public scenario. The model received the scenario file, policy catalog, action catalog, and output schema, but not the hidden oracle. It did not generate validator outcomes, statistical tests, or final correctness labels. The hidden oracle and all quantitative evaluation were produced independently from fixed rules and stored artifacts. No generative-AI-created images are used in this manuscript; the use-case figures are author-defined vector schematics that summarize the proposed workflows.

5.4 Statistical analysis and scope

Exact binomial confidence intervals are reported for action metrics. The full gateway and schema-only baseline are compared on paired mutations with an exact McNemar test. Local validation latency is measured over repeated in-memory validation and parse-plus-validate runs. These timing values exclude LLM inference, controller interaction, topology simulation, radio execution, and physical robot behavior. The evaluation therefore supports claims about benchmark decisions and artifact admission, not end-to-end autonomous operation.

6 Results and Design Implications

6.1 Blind action selection

The model chooses the correct high-level control mode in all 50 scenarios, giving 100% accuracy with a 95% exact interval of 92.89–100%. It selects an oracle-acceptable action in 45 scenarios (90%; interval 78.19–96.67%) and the preferred action in 43 scenarios (86%; interval 73.26–94.18%). The gap between mode and action accuracy matters. The model knows when to act, clarify, or refuse, but choosing the exact network abstraction is harder.

All five unacceptable choices have the same cause: the model uses generic rerouting (A1) when the hidden oracle requires access-path switching (A2). Both actions are bounded and reversible, so a local schema and safety gateway can admit the artifact even though the action family is semantically less precise. This error exposes an interface-design lesson: a catalog must separate forwarding paths from access technologies and make illegal cross-level parameter combinations unrepresentable.

6.2 Mutation detection and statistical significance

The schema-only baseline detects 150 of 400 corruptions (37.5%): all unknown-action, name-mismatch, and missing-field defects. It misses fabricated evidence, undeclared resources, invalid bounds, rollback corruption, and risk mismatch. The full gateway detects all 400 corruptions. The paired comparison has 250 cases caught only by the full gateway and none caught only by schema validation, yielding an exact McNemar p-value of approximately 1.11×10^{-75}. The result is statistically decisive for this mutation set.

Median in-memory validation time is 0.421 ms for the entire 50-artifact batch, or 8.415 microseconds per artifact. The 95th percentile is 0.478 ms. Parse plus validate takes a median of 0.678 ms and a 95th percentile of 0.834 ms. These checks are small compared with model inference or network execution, which supports placing the admission gateway synchronously in an edge-management workflow.

6.3 Implications for stationary applications

Stationary applications benefit from the gateway even when their control deadlines are relaxed. A camera agent may generate a QoS or bitrate request based on loss. Without evidence checks, it can cite an invented queue or claim a target path is available. Without scope checks, it can prioritize all video rather than the active alarm stream. Without rollback, a temporary emergency policy can become permanent.

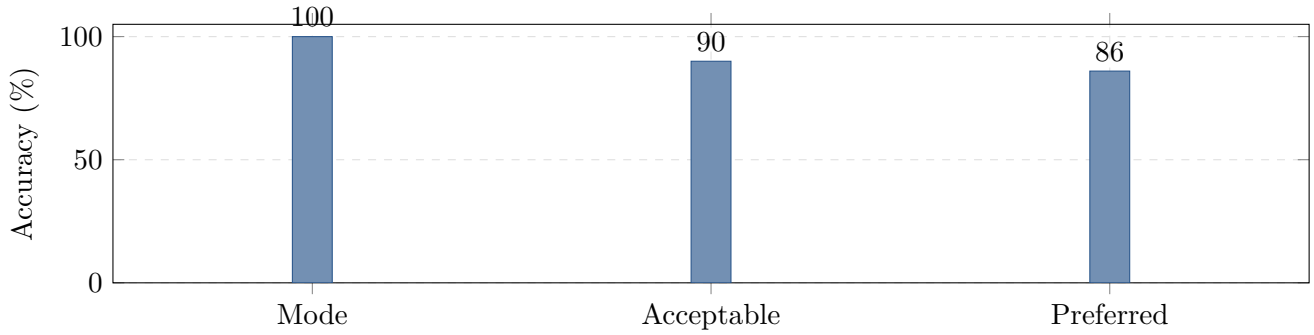

Figure 7: Blind decision and action-selection accuracy on the 50 scenarios

Table 5: Detection by corruption class

Corruption class	Schema only	Full gateway
Unknown action	50	50
Action-name mismatch	50	50
Missing required field	50	50
Fabricated evidence	0	50
Ontology violation	0	50
Invalid bound	0	50
Rollback / semantic error	0	50
Risk mismatch	0	50
Total	150	400

The benchmark result that schema validation misses five of eight corruption classes directly applies to these use cases.

The agent itself should remain specialized. Vision and anomaly models handle continuous perception; an LLM is invoked for incident explanation, policy retrieval, or multi-step coordination. This reduces compute cost and avoids placing a non-deterministic model in a continuous high-rate loop.

6.4 Implications for low-mobility applications

For connected workers, the dominant risk is stale or under-specified context. The benchmark's clarification scenarios show that the model can refuse to invent traffic class, target path, endpoint, or duration. This is important when a worker says "use the better network" or "give me access" without specifying scope. A safe system asks for missing information or monitors longer rather than broadening authority.

Placement should favor the site edge for speech, retrieval, and multimodal reasoning because it offers lower latency and stronger data control than the public cloud. The device should cache validated instructions so that a coverage gap does not force the worker to rely on a partially generated response.

6.5 Implications for high-mobility applications

High-mobility systems add two requirements not exercised by the current benchmark: hard real-time physical control and route-dependent state. The benchmark still validates the network-action boundary, but it is not a robot-motion benchmark. For an AMR, the model may propose moving an inference session or switching an access path; the local controller must continue safe navigation while the request is verified. If verification or execution exceeds the allowed time, the AMR uses its local fallback instead of waiting.

The A1-versus-A2 confusion is more consequential for mobile robots because an access switch can involve authentication, session continuity, security, and edge-service migration, whereas a forwarding reroute may remain within one access domain. Future catalogs should encode these dependencies explicitly and reject a reroute that names an access technology as a generic path.

6.6 Operational cost and automation implications

The industry evidence in the introduction reports lower setup and ongoing costs for many early adopters, but automation can also create new costs: edge accelerators, model lifecycle management, observability, cybersecurity, and specialist integration. The framework reduces avoidable operating cost in three ways. First, specialized local agents filter data and reduce backhaul. Second, mobility-aware placement prevents overprovisioning the cloud for hard real-time tasks that must remain local. Third, typed validation reduces failed changes, emergency rollbacks, and manual incident investigation.

The framework does not assume that every use case needs an LLM. A simple anomaly detector or state machine is cheaper, faster, and safer when the task is narrow. LLMs earn their cost when language, heterogeneous evidence, or multi-step planning is central. This division is consistent with resilience-aware LLM infrastructure and topology-aware workload testing, which show that model performance depends on system placement and resource behavior rather than model quality alone [23–26].

6.7 Benchmark roadmap

The released benchmark provides a reproducible starting point for network-action safety. The next version should add route traces, handover events, compute load, multimodal task records, and executable device-edge-cloud scenarios. Stationary extensions should include video stream degradation and alarm prioritization. Low-mobility extensions should include walking routes, session migration, and retrieval freshness. High-mobility extensions should include AMR trajectories, collision-avoidance deadlines, fleet conflicts, and safe fallback. The same hidden-oracle approach can score agent placement, action family, deadline feasibility, and policy compliance separately.

7 Conclusion and Future Work

Private networks increasingly connect not only devices but also AI decisions that affect industrial operations. A camera, connected worker, and AMR impose fundamentally different mobility and control requirements, so a single generic agent or cloud-first placement strategy is unsafe and inefficient. This paper introduced PrivateAgent, a mobility-aware framework that places deterministic safety and perception on the device, coordination and network intelligence at the site edge, and learning and long-horizon optimization in the cloud. It maps stationary, low-mobility, and high-mobility use cases to agent families, triggers, execution tiers, runtime targets, failure impacts, and benchmark metrics.

The framework also treats every AI-generated network change as an untrusted typed artifact. In the SafeFlow-Real50 evaluation, the model correctly selected act, clarify, or reject in all 50 scenarios, selected an acceptable action in 45, and selected the preferred action in 43. Independent validation detected all 400 injected corruptions, compared with 150 for schema checks, at microsecond-scale local cost. These results do not establish end-to-end robot or private-network autonomy; they establish that the action-admission boundary can be evaluated reproducibly and strengthened substantially without another model call.

Future work will extend the public benchmark with independently authored scenarios, additional models and repeated generations, longer traces, route and handover information, richer RTT and throughput measurements, and human adjudication. The framework will be coupled to a programmable private-network testbed with device, edge, and cloud execution, topology-aware verification, canary rollout, runtime assurance, and measured rollback. AMR experiments will add onboard safety loops, speech and vision tasks, service migration, and fleet coordination. The action catalog will also separate access switching, forwarding, compute placement, and physical-task authority more strictly so that semantically invalid cross-layer actions cannot be represented.

References

[1] Global mobile Suppliers Association. (2025). *Private Mobile Networks December 2025*. `https://gsacom.com/paper/private-mobile-networks-december-2025/`. Accessed 26 July 2026.

[2] Nokia and GlobalData. (2025). *2025 Industrial Digitalization Report*. https://onestore.nokia.com/asset/214958. Accessed 26 July 2026.

[3] Hossain, A. R., et al. (2023). AI-native for 6G core network configuration. *IEEE Networking Letters*, 5(4), 255–259.

[4] Katsaros, K., et al. (2024). AI-native multi-access future networks—The REASON architecture. *IEEE Access*, 12, 178586–178622.

[5] Chen, Y., et al. (2024). NetGPT: An AI-native network architecture for provisioning beyond personalized generative services. *IEEE Network*, 38(6), 404–413.

[6] Tarkoma, S., Morabito, R., & Sauvola, J. (2023). AI-native interconnect framework for integration of large language model technologies in 6G systems. *arXiv preprint arXiv:2311.05842*.

[7] Hamadanian, P., Arzani, B., Fouladi, S., Kakarla, S. K. R., Fonseca, R., Billor, D., Cheema, A., Nkposong, E., & Chandra, R. (2023). A holistic view of AI-driven network incident management. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks* (pp. 180–188). https://doi.org/10.1145/3626111.3628176.

[8] Ma, D., et al. (2025). LORADOCTOR: LLM-driven diagnosis and adaptive policy optimization for reducing packet error rate in LoRaWAN networks. In *Proceedings of the International Conference on Machine Learning and Applications*. https://doi.org/10.1109/ICMLA66185.2025.00181.

[9] Neupane, K., Kostage, K., Peppers, S., Pandey, A., Calyam, P., & Qu, C. (2025). NetPrompt: LLM-driven programmable network policy management and optimization. In *2025 34th International Conference on Computer Communications and Networks*. IEEE. https://doi.org/10.1109/ICCCN65249.2025.11133886.

[10] Huang, L., Wu, Y., & Simeonidou, D. (2026). Optimizing system performance for autonomous recovery in AI-native networks using large language models. *IEEE Transactions on Cognitive Communications and Networking*.

[11] Sahin, O., Sathya, V., & Yavuz, M. (2025). Private networks: Evolution, ecosystem, use cases, architecture, spectrum, and deployment challenges. *Computer Communications*, 242, 108295. https://doi.org/10.1016/j.comcom.2025.108295.

[12] Liu, F., Farkiani, B., & Crowley, P. (2024). A survey on large language models for network operations and management: Applications, techniques, and opportunities. Preprint.

[13] Hong, J., Van Tu, N., & Hong, J. W.-K. (2025). A comprehensive survey on LLM-based network management and operations. *International Journal of Network Management*, 35(6), e70029. https://doi.org/10.1002/nem.70029.

[14] Wei, Y., et al. (2025). Automated network protocol testing with LLM agents. *arXiv preprint arXiv:2510.13248*.

[15] Raptis, N., Fonseca, J. P., Adhane, G., Ramantas, K., & Verikoukis, C. (2026). Reliable LLM-enabled intent-driven management: A validation pipeline for network automation. In *IEEE International Conference on Communications*. IEEE.

[16] Petrovic, N., Krstic, D., & Glabowski, M. (2026). LLM-driven approach for safe and secure network management by design in IoT-based systems. *Symmetry*, 18(2), 337.

[17] Melis, A., et al. (2018). A policy checker approach for secure industrial SDN. In *2018 2nd Cyber Security in Networking Conference*. IEEE.

[18] Bringhenti, D., Yusupov, J., Molina Zarca, A., Valenza, F., Sisto, R., Bernal Bernabe, J., & Skarmeta, A. (2022). Automatic, verifiable and optimized policy-based security enforcement for SDN-aware IoT networks. *Computer Networks*, 213, 109123. https://doi.org/10.1016/j.comnet.2022.109123.

[19] Wang, P., et al. (2025). AgentArmor: Enforcing program analysis on agent runtime trace to defend against prompt injection. *arXiv preprint arXiv:2508.01249*.

[20] Joo, Y., Ribeiro, V., Feldmann, A., Gilbert, A. C., & Willinger, W. (2001). TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations. *ACM SIGCOMM Computer Communication Review*, 31(2), 25–37.

[21] Owezarski, P., & Larrieu, N. (2004). A trace based method for realistic simulation. In *IEEE International Conference on Communications* (Vol. 4, pp. 2099–2103). https://doi.org/10.1109/ICC.2004.1312915.

[22] Raith, P., Rausch, T., Furutanpey, A., & Dustdar, S. (2023). faas-sim: A trace-driven simulation framework for serverless edge computing platforms. *Software: Practice and Experience*, 53(12), 2327–2361.

[23] Karfakis, G., Tahmasebi, F., Chen, B., Yao, L., Mitra, S., Pan, T., Kwon, H., & Gupta, P. (2025). RAPID-LLM: Resilience-aware performance analysis of infrastructure for distributed LLM training and inference. *arXiv preprint arXiv:2512.19606*.

[24] Vogt, F. G., Yang, Z., Lopes, V. H. S., Rodriguez, F., Luizelli, M. C., Rothenberg, C. E., & Papagianni, C. (2025). Poster: Reconstructing LLM training workloads: A topology- and model-aware network tester. In *Proceedings of the 21st International Conference on Emerging Networking Experiments and Technologies* (pp. 41–42). https://doi.org/10.1145/3765515.3771757.

[25] Ding, E., Oh, B., Kataria, B., Guo, K., Gvero, J., Vijaya Kumar, A., Devraj, A., Bowen, L., Sonwane, A., Manzoor, E., & Singh, R. (2026). CCL-Bench 1.0: A trace-based benchmark for LLM infrastructure. *arXiv preprint arXiv:2605.06544*.

[26] Zhao, B., He, Y., Yang, G., Chen, S., Liu, R., Zhang, T., & Xu, W. (2025). MegatronApp: Efficient and comprehensive management on distributed LLM training. *arXiv preprint arXiv:2507.19845*.

Statements and Declarations

Funding. The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing Interests. Vanlin Sathya is affiliated with Celona Inc., whose private-network measurement traces contribute to the data analyzed in this study. The other authors declare no relevant financial or non-financial interests.

Author Contributions. Onur Sahin contributed to conceptualization, methodology, software, data curation, formal analysis, visualization, and writing of the original draft and revisions. Vanlin Sathya contributed to conceptualization, industry use-case design, validation, supervision, and manuscript review and editing. Lyutianyang Zhang contributed to methodology, formal analysis, validation, visualization, and manuscript review and editing. All authors read and approved the final manuscript.

Data Availability. The benchmark scenarios, hidden oracle, generated actions, evaluation outputs, and deterministic evaluator are available at https://github.com/OnurSahh/Trace2Action_GitHub_Repo. The fine-grained source traces are not redistributed because they may contain deployment-sensitive location information.