Connectivity Is the Missing Spec for Physical AI

A perspective from Cyberwave and Apeiroon

APEIROON
MOBILE AI INFRASTRUCTURE

cyberwave

Connectivity Is the Missing Spec for Physical AI

Why Scalable Robotics Depends on Mobile AI Infrastructure

A perspective from Cyberwave and Apeiroon

May 2026

Cyberwave

The software infrastructure layer for Physical AI, connecting AI to robots and autonomous systems through digital twins, edge runtimes, fleet orchestration, and a universal robot API.

Apeiroon

Compact, rapidly deployable private 5G network with an integrated edge GPU compute module — Roadwave — for environments where standard connectivity fails. Purpose-built for mission-critical, field-deployed autonomous systems.

Executive Summary

Physical AI moves artificial intelligence from digital systems into machines that sense, decide, and act in the physical world. This shift changes the role of connectivity. Mobile robots, drones, and autonomous systems do not merely need access to a network. They need predictable communication while moving through environments shaped by contention, changing radio conditions, handovers, interference, and operational stress.

As fleets scale, the communications layer becomes a defining part of the system architecture. It determines how shared state is maintained, where inference can run, how machines coordinate, how operators remain informed, and how safely the system degrades when conditions change. Enterprise Wi-Fi and public cellular can support parts of this stack, but mission-critical mobile autonomy requires a more controllable communications substrate.

This paper argues that private mobile networks, combined with edge AI and fleet orchestration, forms a new category of **Mobile AI Infrastructure**: the integrated layer that enables Physical AI in motion. Its role is not simply to connect robots. It is to make connectivity, compute, inference, orchestration, and fallback behaviour co-designable.

The central design question is therefore no longer whether a robot is connected. It is which inference workloads can safely leave the robot, under which network conditions, with what service guarantees, and with what fallback behaviour when those conditions degrade. For Physical AI to scale, connectivity must become explicit, measurable, and engineered as part of the autonomy stack.

Table of Contents

Why we wrote this

Physical AI extends artificial intelligence beyond systems that interpret digital information into systems that sense, decide, and act in the physical world. It encompasses robots, drones, autonomous vehicles, and other intelligent machines whose outputs are not merely predictions or recommendations, but actions with direct physical consequences.

This shift changes the architecture of AI systems. Once intelligence is coupled to movement and action, performance depends not only on model accuracy, but on the reliability of the wider operational stack: onboard compute, control systems, networking, orchestration, cloud services, and human supervision. These layers must exchange information under conditions of mobility, constrained bandwidth, latency sensitivity, safety requirements, and environmental variability. In Physical AI, intelligence is therefore not simply deployed onto infrastructure; it is shaped by the infrastructure that supports it.

This is where conventional assumptions begin to break down.

In early deployments, connectivity is often treated as a background utility. A small number of machines operate in controlled environments, with limited mobility and modest contention for network resources. Under these conditions, enterprise Wi-Fi or public cellular connectivity may appear sufficient. The system functions, and the network remains largely invisible.

At scale, that illusion dissolves.

As fleets expand and operating environments become more dynamic, the network shifts from passive medium to active system constraint. Autonomous machines require continuous mobility support, reliable uplink capacity, bounded latency, controlled jitter, and predictable performance to maintain shared state, coordinate actions, and transmit sensor data. When these conditions are not met, failures are rarely graceful. They appear as discontinuities: control loops destabilise, coordination degrades, observability fragments, and system behaviour becomes increasingly opaque. The underlying problem is structural. The communications layer was never designed as part of the system architecture.

This paper is written from a joint perspective. Cyberwave develops the software infrastructure layer for Physical AI, including digital twins, edge AI runtimes, fleet orchestration, and hardware abstraction. Apeiroon develops Roadwave, a family of compact, rapidly deployable private 5G networks with integrated edge GPU compute to host AI inference and orchestration — for autonomous machines operating in the field. From these two vantage points — software infrastructure and private connectivity — the same conclusion emerges: reliable Physical AI at scale requires communications infrastructure

to be treated as a first-order architectural component, co-designed with compute, inference, and control, rather than assumed as a background service.

The scope of this paper is deliberately focused. A fixed industrial robot, a stationary inspection system, and a mobile autonomous fleet all belong to the broader category of Physical AI, but they do not impose the same infrastructure requirements. The most demanding scaling challenge arises when intelligence must remain coordinated while machines move through real, changing environments.

Physical AI is therefore the broader category: intelligent systems that sense, decide, and act in the physical world. Physical AI in motion is the more demanding subset: robots, drones, autonomous vehicles, and other mobile machines that must remain coordinated while navigating dynamic environments. We refer to the infrastructure required for this class of systems as **Mobile AI Infrastructure**: the integrated stack of private wireless connectivity, edge inference, orchestration, observability, and lifecycle management required to operate intelligent machines in motion.

The analysis that follows proceeds in three parts. First, it examines why best-effort connectivity assumptions fail as autonomous systems scale. Second, it outlines an architectural framework for distributing intelligence across device, edge, network, orchestration, and cloud layers. Third, it identifies the technical and operational questions that arise in real deployments, where system performance is determined not by individual components, but by the behaviour of the system as a whole.

1. The problem with assuming connectivity

Enterprise Wi-Fi and public 5G can each support elements of the Physical AI stack. Neither, however, was designed around the combined requirements of dense fleets of mobile, sensor-rich machines operating in shared and dynamic environments. As robotics moves from controlled pilots to production-scale deployment, this mismatch becomes structural.

The challenge is not simply network capacity. Modern wireless systems can deliver impressive peak throughput. The more consequential question is whether they can sustain predictable communication under mobility, contention, and coordination load, while maintaining stable links to edge compute, orchestration systems, and human operators. In Physical AI, system performance depends less on headline bandwidth than on communication consistency: bounded latency, controlled jitter, predictable packet-loss behaviour, session continuity, and rapid recovery from degradation.

A single robot can often tolerate transient connectivity problems because the impact remains local. A fleet changes the failure mode. Once multiple robots share space, tasks, and situational awareness, the network becomes part of the system's operating logic. It

carries not only data, but state, intent, constraints, and safety-relevant events. When communication becomes unstable, the result is not merely a weaker connection. Coordination degrades, observability fragments, and the system is forced into a more conservative, less efficient, or less transparent mode of operation.

1.1 What changes when robots scale

Scaling a robotic system is not a linear extension of a single-device deployment. Moving from one robot to many introduces qualitative changes in system behaviour.

First, aggregate data demand rises. Each robot contributes sensor streams, telemetry, logs, and control traffic. Even with aggressive onboard filtering, cumulative load grows quickly and begins to compete for shared network resources.

Second, coordination traffic emerges as a distinct requirement. Robots operating in shared space must exchange state, intent, priorities, and constraints. These messages are often modest in volume, but highly time-sensitive. They are largely absent in isolated single-robot deployments, yet become central to multi-agent operation.

Third, connectivity becomes a shared dependency. With one robot, a brief communication loss is typically local. In a fleet, it can propagate across the system. A dropout may interrupt coordination, degrade shared situational awareness, and, in tightly coupled workflows, trigger fleet-wide fallback behaviour.

These effects are not anomalies. They are intrinsic to distributed cyber-physical systems. Addressing them requires connectivity to be designed as part of the operating architecture, rather than assumed to be sufficient by default.

1.2 The mobility problem

Mobile robots do not operate from fixed points. They move continuously through environments with heterogeneous, spatially uneven, and time-varying radio conditions. Connectivity must therefore be maintained not at a location, but along a trajectory.

This introduces constraints that differ fundamentally from conventional client connectivity. Signal quality varies with position, orientation, speed, and environmental change. Obstructions, reflections, metal structures, moving vehicles, and interference patterns reshape the radio environment in real time. Transitions between coverage domains — whether Wi-Fi access points or cellular cells — introduce handover events that can disrupt communication if not tightly controlled.

Static network planning rarely captures these dynamics. A site survey may indicate adequate coverage, yet communication quality deteriorates once machines move at operational speeds through real workflows. Handoffs that are acceptable for human-facing

applications, such as browsing or video streaming, can still disrupt time-sensitive robotic communication, where delayed telemetry, stalled commands, or short gaps in state synchronization affect coordination and control.

Public cellular networks provide strong wide-area mobility, but offer limited control over local performance characteristics. Enterprise Wi-Fi can be optimized locally, but often struggles to deliver seamless mobility under load, particularly in RF-complex industrial environments.

In field operations — construction sites, logistics yards, temporary industrial zones, and emergency-response settings — the challenge becomes more acute. Infrastructure may be temporary, incomplete, or absent. Connectivity must therefore be deployable, relocatable, and adaptable as the operational footprint changes. A static network is poorly matched to a dynamic system.

Mobility, in this context, is not an optional feature. It defines the operating conditions under which the system must function and remain reliable.

1.3 The data profile at fleet scale

Autonomous machines generate substantial data even when designed for efficiency. Camera feeds, LiDAR outputs, machine telemetry, localization data, health signals, and state updates create continuous flows of information. Much of this can and should be processed locally, but a meaningful fraction must still be transmitted for coordination, monitoring, remote intervention, model improvement, and lifecycle management.

Per robot, this may amount to tens of megabits per second, depending on sensor configuration, control architecture, and operating mode. At fleet scale, aggregate demand can reach gigabit-class levels once perception data, coordination traffic, model distribution, logging, and management overhead are combined.

The challenge is not raw capacity alone, but traffic structure. Robotic workloads combine steady telemetry, bursty perception uploads, low-volume but time-critical coordination messages, teleoperation streams, and occasional bulk transfers. These flows have different operational priorities and cannot be treated equivalently. A software update can wait. A degraded video stream may be acceptable. A delayed coordination event may not be.

Conventional access networks are rarely specified around this traffic profile. They are typically optimized for aggregate throughput or average user experience, not for preserving performance guarantees across heterogeneous, time-sensitive flows under mobility and contention.

The implication is straightforward: data volumes are manageable, but only when explicitly engineered for. Without that, variability — not peak load — dominates system behaviour.

1.4 Coordination is a network problem

In multi-robot systems, the critical connectivity challenge is not data transport alone. It is coordination.

Robots sharing physical space must exchange position, intent, task progress, priorities, safety events, and operational constraints through an orchestration layer. These messages are usually small, but their value decays rapidly with delay. A stale position update, a delayed yield signal, or a missed synchronization event can reduce throughput, create deadlock, trigger conservative behaviour, or, in safety-critical contexts, raise the risk of hazardous interactions.

This reframes the definition of connectivity. It is no longer sufficient for a robot to be nominally "connected." What matters is whether communication can be maintained within defined bounds of latency, loss, and recovery while the machine moves through the environment.

From this perspective, connectivity becomes part of the system's control surface. It shapes how coordination algorithms behave, what assumptions they can safely make, and what performance guarantees they can realistically provide.

1.5 The coverage continuity requirement

Most access networks partition space into coverage regions. Movement across these regions introduces handoffs. Each handoff is a potential discontinuity.

If transitions are not sufficiently fast, predictable, and loss-tolerant, the orchestration layer experiences gaps in telemetry or delays in command delivery. Viewed in isolation, these interruptions may appear minor. In coordinated operation, they often occur precisely when state consistency is most important: at crossings, merges, handover points, and moments of shared task execution.

In industrial environments — warehouses, outdoor yards, ports, logistics hubs, and field operations — these are not merely QoS issues. They are failures in the control path.

A robust Physical AI architecture therefore assumes that connectivity degradation will occur and defines how machine behaviour should adapt. In practice, this requires a layered control hierarchy: safety-critical functions remain local, while network-dependent behaviours degrade gracefully — from full coordination, to reduced interaction, to safe autonomous fallback.

These transitions must be explicitly defined, implemented, and validated under real operating conditions. They cannot be inferred from nominal coverage maps or laboratory network benchmarks.

1.6 Failure modes of underspecified connectivity

In early deployments, connectivity issues are often tolerated as intermittent operational friction. At scale, the same issues become system-level constraints.

Typical failure modes include:

- roaming interruptions during movement between coverage zones
- transient loss of shared state during coordinated tasks
- congestion between competing traffic classes (perception, telemetry, control)
- degraded latency under peak fleet activity
- cascading mission interruptions triggered by short but frequent dropouts
- loss of observability when telemetry degrades under fault conditions

These effects are not rare. They are often invisible in pilots because pilots rarely reproduce the conditions that matter most: higher device density, sustained traffic load, continuous mobility, environmental variability, and operational pressure.

Their significance lies in timing. They emerge precisely when robotic systems are expected to become dependable infrastructure: at scale, under load, and in motion.

2. Why private mobile networks matter

Private mobile networks are dedicated cellular networks — 4G, 5G, and eventually 6G — deployed for a specific site, fleet, or operational environment. Unlike shared public infrastructure, they give operators direct control over coverage, capacity, access, security, traffic policies, and quality of service.[1]

For Physical AI, this matters because mobile autonomy depends on predictable communication under motion, contention, and operational stress. Private mobile networks allow coverage, mobility, device identity, traffic prioritisation, local breakout, and edge integration to be engineered as part of the system architecture, rather than inherited from best-effort infrastructure.

Their value is therefore not peak bandwidth alone, but architectural control.

[1] *In 3GPP terminology, private mobile networks correspond to Non-Public Networks (NPN), with two deployment models — Standalone NPN (SNPN) and Public Network-Integrated NPN (PNI-NPN). Several requirements developed in this paper map to standardised 3GPP functions: runtime network-state exposure to the Network Exposure Function (NEF) and Network Data Analytics Function (NWDAF); traffic differentiation to 5QI-based QoS handling; time-sensitive coordination to the 5G TSC framework (Rel-16 onwards), including TSN integration for deterministic industrial bridging. This paper does not attempt to expand on these standards; it addresses the system-level architecture for which they provide one set of building blocks.*

This does not imply that private mobile networks replace every other connectivity technology. Wired industrial Ethernet remains the right choice for deterministic fixed cells and stationary equipment. Wi-Fi remains useful for high-throughput local connectivity where mobility, interference tolerance, and service guarantees are less critical. Public 5G remains valuable for wide-area reach and operator-managed coverage.

The point is narrower, and more important: when fleets of intelligent machines must remain coordinated while moving through real environments, private mobile networks provide a more controllable foundation for treating communications as an integral part of the Physical AI system architecture.

Connectivity	Strengths	Limitations for mobile Physical AI
Enterprise Wi-Fi	Familiar, cost-efficient, high indoor throughput, broad device ecosystem.	Roaming continuity, interference management, deterministic QoS, large-area mobility and mission-critical continuity can be difficult under fleet load.
Public 5G	Wide-area coverage, native mobility support, mature operator infrastructure.	Limited local control over site-specific QoS, on-premise autonomy, sovereignty, and service policies; dependence on operator coverage and network configuration.
Private mobile networks	Local control, mobility, QoS, isolation, device identity, local breakout, and edge integration.	Requires spectrum strategy, radio planning, device integration, operational competence and empirical validation in the target environment.
Wired industrial networks	Excellent determinism, reliability, and performance for fixed equipment and machine cells.	Unsuitable for untethered mobile robots, drones, autonomous vehicles, or rapidly changing field environments.

3. The AI inference challenge

Physical AI changes both the economics and the architecture of inference. In digital AI, inference is commonly delivered as a cloud service: a request is transmitted, a model processes it, and a result is returned. Latency matters, but primarily as a matter of responsiveness and user experience.

In Physical AI, inference is coupled directly to movement. A perception output may determine whether a robot slows down, yields, reroutes, grasps, stops, or requests human intervention. The cost of delay is therefore no longer mere inconvenience. It can mean degraded coordination, lower throughput, reduced resilience, or unsafe behaviour.

This creates an inference-placement problem. Some workloads must remain onboard the machine, particularly those associated with safety, balance, collision avoidance, and reflex-like responses. Other workloads can be executed at the local edge, where larger models, richer scene understanding, shared world models, and fleet-level optimisation become feasible. Still others belong in the cloud, where latency is less critical and global context, historical data, and large-scale computation create the greatest value.

The boundary between these domains is not fixed. It depends on mission, robot capability, model size, sensor data rate, network conditions, and acceptable level of operational risk. A system that can offload perception in one zone may need to revert to onboard inference in another. A fleet that relies on shared edge inference during normal operation may need to degrade gracefully during congestion, handover events, or partial network failure.

Private mobile networks matter because they make this boundary more controllable. By combining local coverage, quality-of-service controls, mobility management, network exposure, and integration with edge compute, they allow inference workloads to be distributed deliberately rather than opportunistically.

Inference location	Appropriate workloads	Connectivity implication
Onboard robot	Safety reflexes, low-level control, collision avoidance, balance, essential perception.	Must function without network continuity; loss of connectivity must not create immediate unsafe behaviour.
Edge compute / private mobile network site	Shared perception, scene understanding, model ensembles, fleet coordination, teleoperation support, and larger inference workloads.	Requires bounded latency, sufficient uplink capacity, QoS policy, session continuity, and runtime visibility into network state.
Cloud	Training, fleet analytics, historical logs, compliance, global optimization, model lifecycle management.	Should not sit in the real-time control path; cloud degradation must not interrupt safe local operation.

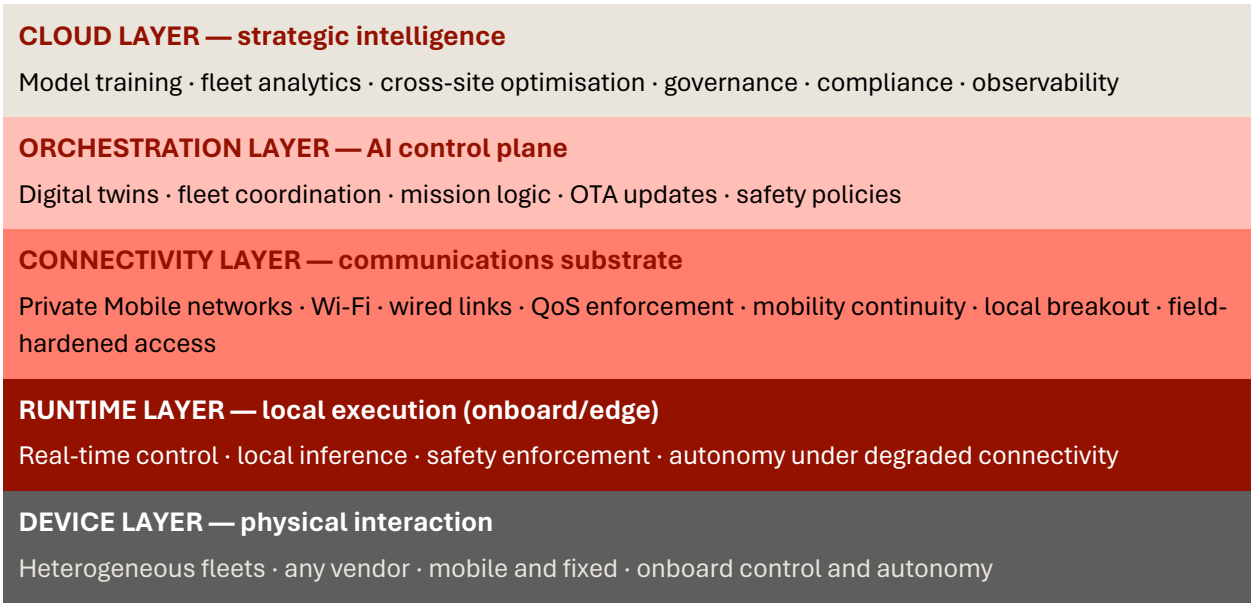
4. The architecture

A Physical AI system is not a monolith. It is a distributed architecture in which intelligence is partitioned across layers, each operating under different constraints of latency, bandwidth, reliability, and autonomy. System behaviour emerges not only from the capabilities of each layer, but from the quality of the interfaces between them.

In practice, five layers can be distinguished. Each layer has a distinct role, but one is consistently under-specified: connectivity.

Compute is provisioned. Models are optimized. Control logic is carefully structured. The network, by contrast, is often treated as an external service assumed to be "good enough." In Physical AI, that assumption does not hold under real operating conditions.

The network is not merely a transport mechanism. It defines the system's operating envelope: how quickly state can propagate, how reliably machines can coordinate, how safely the system degrades under failure, and where intelligence can be placed across device, edge, and cloud. In practical terms, it becomes part of the control architecture.



The sections below describe the four layers where Cyberwave and Apeiroon operate — Runtime, Connectivity, Orchestration and Cloud — working outward from the point of action. The Device Layer, while foundational, is treated here as given: a heterogeneous collection of robots, sensors, and machines that interface with the physical world. The role of the remaining layers is to connect intelligence to that world reliably, despite its variability.

4.1 The Runtime Layer — execution under constraints

The Runtime Layer is where time-critical workloads are executed. Located on or near the operational site, it supports model inference, local control loops, safety enforcement, telemetry ingestion, and coordination functions that cannot depend on cloud round-trips.

In Cyberwave's architecture, these functions run in a lightweight edge runtime deployed either onboard the machine or on edge compute. The cloud remains important for workspace and workflow management, data storage, analytics, and enterprise integration.

However, real-time operation is designed to continue when cloud connectivity is degraded or unavailable.

Conceptually, the system maintains a bidirectional coupling between physical and virtual state. Each robot is represented by a digital twin in software. Telemetry continuously updates the virtual representation of the machine; commands, policies, and mission assignments flow back to the physical asset. Coordination is structured across mission, coordination, execution, and safety layers, keeping planning, traffic management, actuation, and safety enforcement logically distinct.

This edge-first design has direct implications for connectivity. Time-critical execution remains local, but the Runtime Layer still depends on the network for coordination, telemetry, observability, teleoperation, and lifecycle management. Its communications requirements are therefore best understood by traffic class.

Traffic class	Typical content	Primary network requirement
Control and safety	Motion commands, stop signals, safety interlocks.	Ultra-low latency, tightly bounded jitter, highest QoS priority, local fallback.
Coordination and shared state	Position, intent, task progress, yield/merge events, shared constraints.	Low latency, high continuity, fast recovery across handoffs.
Telemetry and observability	Health metrics, logs, status updates, diagnostics.	Continuous delivery, low loss, timestamp consistency.
Perception and teleoperation	Video, images, point clouds, high-rate sensor streams.	High sustained throughput, adaptive bitrate, graceful degradation.
Bulk data and lifecycle	OTA updates, model pushes, maps, historical logs.	Burst-tolerant, schedulable, lowest priority.

4.2 The Connectivity Layer — the system constraint

The Connectivity Layer is often described as technology-agnostic. In practice, it is tightly constrained by the operating requirements of the system it supports. Once those requirements are made explicit — mobility continuity, local autonomy, traffic prioritisation, isolation, and predictable service quality — the set of viable architectures narrows substantially.

The network defines the practical limits of distributed intelligence. It determines how rapidly state can propagate, how reliably machines can coordinate, how safely behaviours degrade during disruption, and where inference can be placed across device, edge, and cloud.

Five requirements deserve explicit attention because they frequently determine whether Physical AI deployments remain robust beyond the pilot stage:

- Bounded jitter and packet-loss budgets. Mean latency is insufficient; tail behaviour matters.
- Explicit uplink capacity. Robotics workloads are often uplink-heavy because video, telemetry, and sensor summaries flow from machine to edge.
- Time synchronization. Shared world models require accurate event ordering across robots, cameras, sensors, and edge services.
- Session continuity during mobility. MQTT, WebRTC, REST and control-plane sessions must either survive roaming events or recover quickly enough that orchestration confidence is not lost.
- Runtime visibility into network state. Signal quality, cell load, congestion, handover events, and QoS state should be exposed to orchestration and edge runtimes.

These requirements shift connectivity from an IT consideration to a systems-engineering concern. The network must be designed, monitored, and validated as part of the operational architecture.

Apeiroon's Roadwave is one such platform: a self-contained private 4G/5G deployment that integrates radio access, mobile core, and edge compute into compact, field-deployable units — and exposes runtime network state to the orchestration layer through standard interfaces.

4.3 The Orchestration Layer — maintaining coherence

The Orchestration Layer is the control plane of the system. It maintains a coherent view of the fleet and directs its behaviour: assigning tasks, resolving conflicts, sequencing actions, enforcing policies, and updating system state.

This layer transforms a collection of autonomous machines into a coordinated operational system. It depends on the Runtime Layer for local execution and on the Connectivity Layer for maintaining a timely, sufficiently consistent representation of the world.

Its central challenge is not merely computation, but coherence: ensuring that the system's internal model remains aligned with physical reality. If telemetry becomes stale, if a handover interrupts shared state, or if the orchestration layer lacks visibility into deteriorating network conditions, it may continue to reason over a world that has already changed.

While orchestration can be cloud-hosted, co-locating it with edge and connectivity infrastructure reduces coordination latency and supports operation in disconnected,

sovereign, or bandwidth-constrained environments. In many industrial and field settings, this is not a marginal optimization. It is a precondition for dependable operation.

4.4 The Cloud Layer — global context, outside the real-time control loop

The Cloud Layer handles functions that benefit from scale, persistence, and global context, but do not belong in the real-time control path. These include model training on aggregated fleet data, cross-site analytics, compliance logging, asset management, dataset management, and enterprise integration.

A well-designed Physical AI deployment minimises unnecessary dependence on the cloud during operation. The edge filters, aggregates, and prioritises data before upload. Raw sensor data remains local where appropriate, or is discarded after processing if it has no enduring operational or analytical value.

Cyberwave's architecture partitions responsibilities accordingly. Time-critical execution, robot-facing integration, and local control remain at the edge. Real-time commands and telemetry are exchanged through lightweight messaging such as MQTT. Teleoperation video is optimised for direct low-latency streaming. The cloud supports asset management, permissions, workflows, datasets, training runs, historical telemetry, mission logs, event streaming, and enterprise integration.

The guiding design principle is straightforward:

Keep control local. Keep coordination near the edge. Use the cloud for aggregation, learning, and system-wide optimisation.

5. Deployment archetypes

The architectural role of a private mobile network is determined not simply by whether robots are connected, but by which functions in the autonomy stack become dependent on that connection.

Three functions are especially important. Compute refers to the processing resources available to execute workloads. Inference refers to the execution of AI models over sensor, telemetry, or state data. Control refers to the generation of decisions and commands that shape machine behaviour, ranging from supervisory coordination to time-sensitive, control-relevant execution. These functions are closely related in practice, but they should not be conflated. Their placement determines whether the network is merely supportive, operationally enabling, or part of the machine's safety-relevant architecture.

For clarity, this paper distinguishes between onboard compute and edge compute. Onboard compute denotes processing resources physically integrated into, or directly attached to,

the robot. Edge compute denotes nearby offboard processing resources, typically located within the facility or local operational domain and accessed through a private wired or wireless network. The distinction matters because it changes the robot's dependency structure. A robot that performs inference and control locally can tolerate temporary network degradation very differently from one that depends on an external execution environment for control-relevant functions.

This leads to three deployment archetypes.

In the first, the network carries telemetry, fleet coordination, and supervisory data while inference and control remain onboard. In the second, the network becomes part of the inference path by connecting the robot to nearby edge compute, but essential autonomy and safety behaviour remain local. In the third, the network carries control-relevant execution or inference outputs required for control-relevant operation, making it part of the real-time control architecture itself.

These archetypes are not rigid categories. Real deployments may combine elements of all three, and the same robotic system may shift between them depending on task, location, network state, or operating mode. Their value lies in clarifying the central design question:

What function does the network perform in the robot's autonomy stack?

That question determines the latency budget, the acceptable failure modes, the validation burden, and ultimately the safety case of the deployment.

5.1 Configuration A — onboard inference

In the first configuration, inference and control runtimes are located onboard the robot. A dedicated compute unit — for example, an onboard AI module with integrated GPU and private-network connectivity — is physically integrated into, or directly attached to, the robot through Ethernet or an equivalent local interface. Perception, inference, safety logic, and low-level control all execute locally.

The private mobile network supports telemetry, fleet coordination, observability, over-the-air updates, cloud synchronization, and potentially supervisory remote monitoring. However, it does not sit inside the inference path or the real-time control loop. If connectivity temporarily degrades, the robot continues operating within its local autonomy and safety envelope.

This is the most resilient configuration. It is well suited to warehouse AMRs, inspection robots, cleaning robots, field machines, and rehabilitation systems that must remain safe and functional without continuous network dependence. In such deployments, private mobile networks improve mobility, supervision, and fleet management, while obstacle avoidance, braking, and immediate task execution remain onboard.

The key architectural property of Configuration A is therefore local functional autonomy: the network improves the system, but the robot does not rely on it for core perception, inference, or control.

5.2 Configuration B — edge-assisted inference

In the second configuration, the robot retains local autonomy and safety behaviour, while selected inference workloads are offloaded to edge compute. This edge infrastructure may be deployed within the facility and, in some implementations, co-located with private mobile network functions and GPU resources.

The robot maintains sufficient onboard intelligence for partial inference workloads, local control, and safe operation. The edge runtime augments that autonomy with heavier, richer, or shared AI capabilities.

This architecture is valuable when workloads such as multi-camera scene understanding, shared world-model updates, anomaly detection, pallet recognition, or larger AI models exceed what is practical onboard, yet still require lower latency and tighter operational integration than a distant cloud can provide.

Here, the private mobile network becomes the low-latency path to assisted inference. Its latency, jitter, uplink capacity, and handover behaviour affect the timeliness and usefulness of those offboard outputs. However, because essential safety functions, local navigation, and low-level control remain onboard, the robot can continue operating within a reduced but safe capability envelope if connectivity weakens.

Typical examples include warehouse robots that navigate locally but offload pallet recognition or shared mapping to edge compute, and factory robots that use external vision models for richer scene interpretation. In these cases, offboard intelligence improves what the robot can perceive, predict, or optimize, but it does not replace the robot's core autonomy.

5.3 Configuration C — networked inference & control

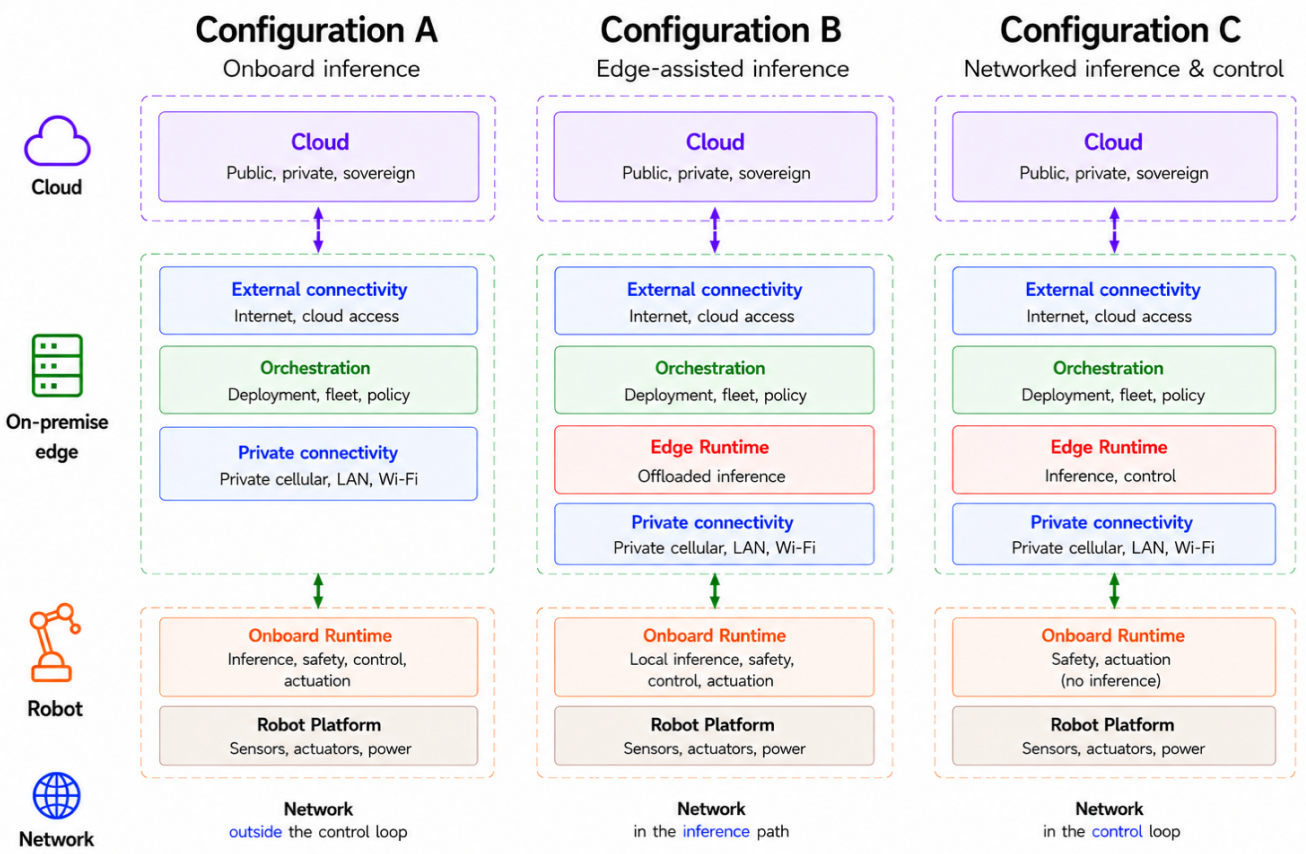
In the third configuration, inference and control-relevant execution are substantially located outside the robot, typically in edge compute co-located with the private mobile network. The robot carries only minimal onboard compute and relies on the network to reach this external execution environment. This can reduce device cost, simplify fleet hardware, and centralize model deployment and software updates.

The trade-off is significant. Once control decisions — or inference outputs required inside the control loop — traverse the network, the private mobile network becomes part of the

control architecture itself. Latency, jitter, packet loss, and handover behaviour directly affect control stability, system responsiveness, and operational safety.

This configuration is therefore appropriate only in tightly bounded environments with explicit latency budgets, validated handover behaviour, conservative safety envelopes, and local fallback states. Examples include low-speed robotic carts in controlled areas, supervised teleoperation, automated yard systems, or demonstrations in which the robot can safely pause when connectivity degrades.

Even in this model, minimal onboard compute remains necessary for I/O handling, emergency stop, safe pause, and fallback behaviour. Unlike Configuration B, where offboard intelligence augments local autonomy, Configuration C externalizes intelligence that the robot depends on for control-relevant operation. It is therefore not merely a compute-placement choice; it is a safety-case decision.



5.4 Why the distinction matters

The three archetypes imply three very different roles for the network.

In Configuration A, the private mobile network supports coordination and supervision while the robot remains locally autonomous. In Configuration B, it enables selective offloading of

inference to nearby edge resources while local safety and autonomy remain intact. In Configuration C, it becomes part of the execution path that shapes control-relevant machine behaviour.

These distinctions should guide both system architecture and validation strategy. A network that carries telemetry, fleet status, and mission updates can tolerate short interruptions that would be unacceptable in a deployment where it carries time-sensitive inference outputs. A deployment in which the network carries control-relevant execution requires stricter latency assumptions, stronger resilience guarantees, explicit fallback behaviour, and a substantially more demanding safety case.

The relevant question is therefore not simply whether a private mobile network is available, nor whether its headline bandwidth is sufficient. The more important question is:

What dependency does the autonomy stack place on the network?

Is the network supervising the robot? Assisting inference? Coordinating a fleet? Carrying human teleoperation signals? Hosting control-relevant execution? Each answer implies a different architecture, a different validation process, and a different risk profile.

The three configurations above are intentionally simple. They do not exhaust the design space, and real deployments may combine elements of all three depending on task, location, network state, and operational mode. Their value lies elsewhere: they make explicit that connectivity is not a generic infrastructure feature, but an architectural choice with consequences for autonomy, resilience, and safety.

5.5 Mobility and handover requirements

Mobility performance matters in all three configurations, but the consequences of interruption differ.

In **Configuration A**, handover events may affect coordination, telemetry, or supervision, but local control continues onboard the robot.

In **Configuration B**, handover interruptions can delay edge-assisted inference and reduce coordination confidence, but the robot should remain safe through onboard fallback behaviour.

In **Configuration C**, handover events can directly affect the execution path and must therefore be treated as control-system events.

As a practical design target, handover interruptions for control- and coordination-relevant traffic should remain below 50 ms wherever network-dependent behaviour is involved.

Indicative thresholds are:

- **< 50ms:** effectively seamless; no application-visible impact on control or coordination.
- **50–100 ms:** degraded but manageable, provided local control or buffering can bridge the gap.
- **100–250 ms:** coordination confidence degrades; the system should transition to conservative behaviour, such as reduced speed, larger separation margins, or reduced task complexity.
- **> 250 ms:** sustained interruption; the system should enter a safe state, pause, or continue only within a bounded autonomous mode.

These thresholds should not be treated as universal guarantees. They depend on robot speed, task criticality, control-loop design, autonomy level, sensor configuration, and safety case. Their purpose is to make the architectural point explicit: as more inference and control move from the robot to the edge, the network must be specified, tested, and validated as part of the system — not after it.

5.6 Design principle

The guiding principle is simple: **Keep safety-critical control local unless the network path has been engineered and validated as part of the control system.**

Private mobile networks make edge-assisted and network-coupled Physical AI more viable because they provide local coverage, mobility management, QoS, isolation, and edge integration. But they do not eliminate the need for robust autonomy, fallback behaviour, and empirical validation.

Their strategic value is more precise: they allow connectivity, compute, inference, and orchestration to be designed together as a single operational architecture, rather than assembled opportunistically after the robot has already been built.

6. Open questions

This paper defines a direction, not a completed architecture. Its central argument is that private mobile networks, edge inference, and orchestration cannot be treated as separate infrastructure domains if mobile Physical AI is to scale beyond pilots and early deployments. They must be designed, instrumented, and validated as a single operating system for machines in motion.

Yet the limits of that system cannot be resolved in abstraction. They must be established empirically, in deployment.

Several questions therefore remain open. They are not secondary implementation details. They define whether **Mobile AI Infrastructure** can become a dependable operating layer for robots, drones, and autonomous machines moving through real environments.

6.1 Network-aware orchestration

The first open question is whether orchestration systems can become genuinely aware of the underlying network.

Most robot orchestration platforms treat the network as an external dependency. Communication is assumed to be available or unavailable, healthy or degraded. For mobile Physical AI, this binary view is too coarse. A fleet moving through a real environment needs operational intelligence about the network itself: signal quality, cell load, handover events, bandwidth availability, QoS state, latency trends, and predicted degradation along the robot's path.

If this information is exposed to the orchestration layer in real time, new behaviours become possible. A robot could slow before entering a weaker coverage zone. Model weights could be pre-positioned before bandwidth falls. Video bitrate could be reduced during congestion. Safety margins could widen when coordination confidence decreases. Time-sensitive traffic could be routed through higher-quality paths.

In this model, the network is no longer passive transport. It becomes part of the fleet's decision context.

Cyberwave already contains several building blocks for this direction: REST-based historical access, MQTT-based real-time telemetry and command delivery, an observability pipeline, and workflow logic that can respond to runtime events. Roadwave exposes complementary network-side data through its standalone private 5G core: per-cell load, link quality, handover events, QoS state, and performance indicators.

The open question is how to formalize the interface between these layers. The challenge is not merely to expose more telemetry, but to define network-state data with the timing, semantics, reliability, and trustworthiness required for control-plane decisions. A mature **Mobile AI Infrastructure** stack should allow orchestration software to reason not only about robot state, but also about the network conditions under which that state is being produced.

6.2 Inference placement

The second open question concerns where inference should run.

In cloud AI, inference placement is mainly an infrastructure and cost decision. In Physical AI, it is also a safety and control decision. A perception result may determine whether a robot

slows down, yields, reroutes, grasps, pauses, or asks for human intervention. The location of inference therefore affects latency, bandwidth, autonomy, and risk.

Some inference workloads must remain onboard the robot, especially those tightly coupled to time-critical local behaviour: obstacle detection, immediate scene interpretation, reflex-like responses, and functions that must continue during network degradation or loss of connectivity. These workloads sit alongside deterministic safety and control mechanisms that cannot depend on external compute.

Other workloads may be better placed at the edge, particularly when they require larger models, shared scene understanding, fleet-level perception, or compute that is impractical to install on every machine. The cloud remains valuable for non-real-time functions such as training, analytics, simulation, cross-site optimisation, lifecycle management, and long-horizon learning.

The boundary between these execution layers should not be fixed once at design time. It should depend on model size, sensor bandwidth, mission risk, available edge compute, radio conditions, robot speed, autonomy level, and the quality of fallback behaviour.

A mature **Mobile AI Infrastructure** stack should therefore support disciplined dynamic inference placement. This does not mean arbitrary offloading whenever edge compute is available. It means moving workloads only when latency budgets, network conditions, safety envelopes, and fallback modes permit it.

The central question is: Which inference workloads can safely leave the robot, under which network conditions, and with what fallback behaviour when those conditions degrade?

6.3 Fleet sizing and network dimensioning

The third open question is how to size the fleet and dimension the network together.

A single autonomous machine produces telemetry, logs, control messages, sensor data, video streams, map updates, diagnostics, and lifecycle traffic. At fleet scale, these flows interact. Average load becomes less important than peak load, burst behaviour, uplink contention, and competition between traffic classes.

The relevant questions include: How does aggregate uplink traffic grow as the fleet expands? How do perception streams interact with coordination messages? What happens when OTA updates, map distribution, or model pushes coincide with peak robot activity? How do handover events affect coordination confidence under load? How much network headroom is required for safe degradation rather than abrupt failure?

These values cannot be estimated reliably from nominal device specifications. They must be measured under realistic conditions: mobility, contention, interference, mixed traffic, sustained operation, and failure events.

Cyberwave captures telemetry across robots, missions, edge runtime behaviour, and orchestration logic. Apeiroon instruments link quality, handover behaviour, QoS enforcement, cell load, and aggregate network performance. The next step is a jointly instrumented deployment in which these measurements are captured simultaneously and correlated.

The objective is to replace indicative assumptions with empirical traces. For Physical AI, the question is not simply how many robots the network can connect. It is how many robots the system can coordinate safely, observably, and predictably under real operating conditions.

6.4 Service-level specification for Mobile AI Infrastructure

The fourth open question is how to specify the service being delivered.

Robotics operators should not buy connectivity in terms of generic bandwidth. Peak throughput is a poor proxy for operational reliability. A fleet operator needs to understand how the network behaves under mobility, congestion, handover, interference, and failure — and how that behaviour maps to robot performance.

A useful service-level specification for **Mobile AI Infrastructure** should be application-aware. It should define requirements by traffic class and operational consequence, including uplink and downlink capacity, latency distribution, jitter, packet-loss targets, handover interruption time, session continuity, QoS under congestion, time synchronization, edge inference availability, network-state exposure, observability, fallback behaviour, and the test conditions under which the specification was validated.

The key shift is from generic connectivity metrics to operational guarantees. Can coordination traffic stay within its latency budget during peak video upload? Can the fleet remain observable during handover? Can safety-relevant messages retain priority during congestion? Can robots continue safely when edge inference becomes unavailable?

For **Mobile AI Infrastructure**, the service level is not only a network property. It is a system property linking connectivity, compute, orchestration, autonomy, and safety.

6.5 Validation and safety case

The final open question is how private mobile network robotics deployments should be validated before production use.

Network testing alone is insufficient. Throughput, latency, jitter, packet loss, handover interruption, and coverage all matter, but they do not prove that the robotic system remains safe and coherent under real operating conditions. The fleet must be tested as an application, not merely as a set of connected devices.

Validation should include stress testing under realistic mobility, density, interference, congestion, and failure conditions. Robots should move through the operational environment at representative speeds, with representative traffic loads, mission logic, sensor streams, teleoperation scenarios, and fallback behaviours. The system should be tested not only when the network performs well, but when it approaches, exceeds, or temporarily loses its service targets.

The critical questions are: Does the fleet remain coherent when latency rises? Does observability degrade gracefully or disappear abruptly? Does coordination remain stable during handover? Are fallback states triggered early enough? Can the system distinguish between a robot fault, a network fault, and an edge-compute fault? Does the operator have enough visibility to intervene?

This is where **Mobile AI Infrastructure** becomes part of the safety case.

For Physical AI, performance testing must be performed from the application's point of view. The relevant question is not whether the private mobile network is fast in isolation. It is whether the autonomous system remains coordinated, observable, and safe under the conditions it will actually face.

7. Next steps

Taken together, these open questions define the next stage of work. **Mobile AI Infrastructure** will mature only when network state, edge inference, fleet orchestration, and robot behaviour are measured as one coupled system. The next stage is empirical.

The architectural arguments in this document should be tested against live deployments, with joint instrumentation across robot fleets, edge runtime behaviour and private mobile network performance. The goal is not only to refine traffic models and handover requirements, but to identify the practical integration patterns that make Physical AI deployments easier to design, validate and scale.

A first concrete instance of this joint instrumentation will be demonstrated at VivaTech 2026 in Paris, where a Cyberwave-orchestrated multi-robot fleet will operate over a Roadwave private 5G deployment. The objective is not demonstration alone, but measurement: capturing the interaction between coordination logic, edge inference and network behaviour under realistic conditions.

The desired output is a reference architecture for **Mobile AI Infrastructure**: a jointly specified and benchmarked pattern for deploying private mobile networks, edge inference and fleet orchestration as one operational system.

8. Conclusion

Physical AI is emerging as a full-stack systems problem. Advances in models, hardware, and orchestration software are necessary, but not sufficient. Systems fail when the assumptions that bind these components together — particularly around connectivity — remain implicit.

At small scale, connectivity can be treated as a background dependency. At fleet scale, it becomes a defining constraint. It shapes coordination, mobility, safety, and throughput. It determines where intelligence can reside and how reliably the system can operate under real conditions.

The central claim of this paper is therefore direct: connectivity must be treated as a first-class architectural component of Physical AI. It must be engineered for the actual traffic patterns of autonomous fleets, designed for continuity under motion, and integrated with the orchestration systems that depend on it.

Cyberwave and Apeiroon approach this problem from different layers of the stack, but converge on the same conclusion: reliable **Physical AI will require Mobile AI Infrastructure** – software intelligence and communications infrastructure that are not merely compatible, but deliberately co-designed.

An invitation

If you are deploying Physical AI at scale and already confronting the connectivity problem, we would welcome the conversation. This field will advance faster through shared evidence, hard questions, and honest deployment experience than through premature certainty.

Get in touch

Cyberwave · info@cyberwave.com

Apeiroon · info@apeiroon.com

About the authors

Gianluca Verin

Gianluca is a telecommunications engineer with more than 25 years at the frontier of mobile networks. He holds a degree in Electronic Engineering and Telecommunications from the University of Padova and an MSc in Decision Support Systems from the University of Sunderland. After early years at Ericsson, he co-founded Athonet in 2005 — first as CTO, later as CEO — where his team helped pioneer private 4G and 5G, building one of the first commercial private mobile core platforms before the category had a name. Athonet was acquired by Hewlett Packard Enterprise in 2023. He is now co-founder and CEO of Apeiroon, where with the Roadwave product line he is extending private mobile networks from buildings and campuses into vehicles, backpacks, and field operations — the environments where standard infrastructure fails.

Dr. Max Lungarella

Max is a serial deep-tech entrepreneur, robotics engineer, and multidisciplinary researcher with more than two decades of experience at the frontier of artificial intelligence and embodied systems. He holds a degree in Electrical Engineering and a PhD in Natural Sciences from the Artificial Intelligence Laboratory at the University of Zurich. His work spans industrial automation, healthtech, sports engineering, and academic research, with a focus on translating scientific insight into intelligent physical systems that operate in the real world. Max has co-authored more than 100 peer-reviewed publications across AI, robotics, neuroinformatics, biomechanics, embodied intelligence, and complex systems. He has co-founded several ventures, including Cyberwave, where he currently serves as CTO.