



An Efficient Edge Computing Architecture for Smart Industrial Automation Applications

Eirini Belhocine

*Department of Electrical and Computer Engineering, Tonle Sap Institute of Engineering and Commerce,
Cambodia*

eirini.belhocine@tsiec-kh.org

Peer Review Information

*Submission: 02 Jan 2017
Revision: 20 Jan 2017
Acceptance: 05 Feb 2017*

Keywords

*Edge computing, Industrial
Internet of Things, Industry
4.0, smart manufacturing,
real-time automation*

Abstract

Smart industrial automation requires timely processing of sensor, machine, and production data while preserving the deterministic behavior of local control loops. Centralized cloud computing offers elastic resources but can introduce wide-area-network delay, bandwidth consumption, and dependence on remote connectivity. This paper proposes a four-tier edge computing architecture that separates hard real-time control, cell-level preprocessing, plant-level analytics, and enterprise/cloud functions. The methodology combines literature synthesis with a reproducible simulation-based evaluation of three processing strategies: cloud-only, gateway-plus-cloud, and the proposed multi-tier edge model. Tasks are classified by deadline, data volume, and computational demand, then placed at the nearest tier capable of meeting the required service level. A policy engine uses latency budgets, resource availability, and link condition to decide local execution or controlled cloud fallback. In a 5,000-job synthetic industrial workload, the proposed architecture reduced mean response time from 114.4 ms for cloud-only processing to 21.7 ms, increased deadline success from 14.1% to 95.2%, and reduced average WAN traffic from 89.0 KB to 2.7 KB per job. The results indicate that tier-aware placement can improve responsiveness and network efficiency while retaining cloud resources for non-urgent global optimization and historical analysis.

Introduction

Industrial automation is evolving from isolated programmable logic controller (PLC) islands toward cyber-physical production systems in which machines, sensors, robots, supervisory systems, and enterprise applications continuously exchange data. The Industry 4.0 vision therefore depends not only on connectivity but also on the ability to transform data into decisions at the correct time and location. The cyber-physical architecture proposed by Lee, Bagheri, and Kao [1] emphasized layered conversion of physical measurements into cyber-level knowledge, while later industrial communication research

highlighted the need to reconcile Internet-style connectivity with the timing and reliability requirements of automation networks [2].

A cloud-only model is attractive for storage and computational elasticity, but its physical distance from the factory floor creates a structural limitation for latency-sensitive tasks. Wide-area transport, queueing, and remote service variability can be acceptable for enterprise reporting but problematic for visual inspection, condition monitoring, adaptive scheduling, and supervisory control. Fog computing was introduced as an extension of cloud services toward the network edge, with low latency,

location awareness, and support for geographically distributed devices as defining properties [3]. Edge computing subsequently generalized this principle by placing computation and storage close to data sources so that response time, bandwidth cost, privacy, and resilience can be improved [4], [5].

For industrial environments, however, simply moving software from the cloud to a gateway is insufficient. A useful architecture must distinguish hard real-time control from soft real-time analytics, exploit multiple edge tiers, survive intermittent cloud connectivity, and remain compatible with heterogeneous industrial protocols. Software-defined Industrial Internet of Things (IIoT) work demonstrated the value of separating infrastructure, control, and application functions for flexible management of industrial devices and networks [6]. At the same time, emerging edge research showed that proximity can reduce execution delay and provide a practical middle tier between constrained devices and centralized data centers [7], [8].

The objective of this methodology paper is to design and evaluate an efficient multi-tier edge architecture for smart industrial automation using only literature available up to 2017. The proposed design follows four principles: (i) keep safety and deterministic closed-loop control at the device/PLC layer; (ii) perform filtering and fast rule execution at cell gateways; (iii) execute computationally heavier but latency-sensitive analytics at a plant edge cluster; and (iv) use the cloud for long-term storage, cross-site optimization, model training, and enterprise integration. The main research question is whether this tier-aware placement can simultaneously reduce response latency and WAN traffic while increasing the proportion of industrial tasks completed within their deadlines.

Literature Review

The conceptual roots of the proposed architecture come from fog and edge computing. Bonomi et al. [3] described fog computing as an extension of cloud computing to the network edge and identified low latency, mobility, heterogeneity, and streaming/real-time workloads as important characteristics. Yi, Li, and Li [9] later surveyed fog concepts and argued that edge-side resources can address cloud limitations such as unreliable latency and lack of location awareness. These ideas are directly relevant to industrial plants where sensing and actuation are geographically localized and many

decisions lose value if delivered after a process deadline.

Shi et al. [4] formalized edge computing as processing performed near the data source and identified response time, bandwidth savings, privacy, and device constraints as central motivations. Shi and Dustdar [5] similarly emphasized the promise of edge processing for the Internet of Things. Satyanarayanan [7] explained why physical proximity matters for responsiveness, scalability, privacy enforcement, and masking transient cloud outages. Together, these studies justify an architecture that keeps latency-sensitive analytics near machines while still using the cloud when proximity is not essential.

A related line of work used intermediate computing nodes or cloudlets. Satyanarayanan et al. [10] demonstrated the value of a device-cloudlet-cloud hierarchy for applications whose usefulness depends on rapid turnaround. Although their examples were mobile rather than industrial, the architectural lesson is transferable: a nearby resource-rich tier can execute tasks that are too demanding for end devices and too delay-sensitive for a distant cloud. Mobile edge computing surveys by Mach and Becvar [8] and Mao et al. [11] further systematized computation offloading and showed that task placement must jointly consider communication delay, computation demand, and resource allocation.

Industrial literature provides the domain-specific constraints that general edge models do not fully capture. Lee et al. [1] proposed a five-level cyber-physical architecture for Industry 4.0 manufacturing, emphasizing acquisition, conversion, cyber representation, cognition, and configuration. Wan et al. [6] analyzed IIoT communication and proposed a software-defined industrial architecture to improve flexibility and device management. Wollschlaeger, Sauter, and Jasperneite [2] showed that future automation networks must integrate industrial real-time requirements with Internet of Things and Industry 4.0 communication. These studies indicate that edge resources should complement rather than replace PLCs and deterministic industrial networks.

The literature therefore reveals a design gap. Early edge and fog studies established the benefits of proximity, while industrial studies established timing, reliability, and interoperability constraints; however, a concise task-placement methodology that maps industrial workload classes to multiple edge tiers

is still needed. The architecture proposed in this paper addresses that gap by using deadline-aware placement, local fallback rules, and explicit separation of control, analytics, and cloud

functions. All cited sources in this review were published in or before 2017, consistent with the required reference window.

Table 1. Literature synthesis underlying the proposed architecture.

Reference	Main contribution	Use in this methodology
Lee et al. [1]	Layered CPS architecture for Industry 4.0	Separates physical control from cyber-level analytics
Bonomi et al. [3]	Fog computing near data sources	Motivates low-latency, location-aware processing
Shi et al. [4]	Edge-computing vision and challenges	Supports proximity, bandwidth, privacy, and resilience goals
Wan et al. [6]	Software-defined IIoT architecture	Guides industrial device/network management separation
Satyanarayanan [7]	Why proximity matters at the edge	Supports responsive local execution and outage masking
Mach & Becvar [8] / Mao et al. [11]	Computation-offloading and MEC resource models	Supports latency- and resource-aware placement decisions

Methodology

Step 1: Define the industrial workload and timing classes

The first step is to classify each incoming job j by four attributes: input size D_j , estimated compute demand C_j , deadline L_j , and control criticality S_j . Three workload classes are used in the evaluation. Control-support jobs include fast alarms, state estimation, and interlock-related supervisory logic with a 40 ms deadline; these tasks do not replace safety-certified PLC loops. Vision jobs include defect detection and machine-vision inference with an 80 ms deadline. Monitoring jobs include vibration, temperature, power, and condition-analysis events with a 100 ms deadline. This classification makes the placement decision explicit and avoids treating every IIoT message as an identical cloud transaction.

Step 2: Construct the four-tier architecture

The second step is to allocate functions across four tiers. Tier 1 contains sensors, PLCs, robots, drives, cameras, and actuators. It performs deterministic cyclic control and safety functions locally. Tier 2 is the cell/gateway edge and provides protocol translation, timestamping, filtering, feature extraction, caching, and simple rules. Tier 3 is the plant edge cluster, which hosts containerized analytics, machine-vision inference, short-horizon scheduling, local databases, and coordination services. Tier 4 is the cloud/enterprise layer for historical analytics, enterprise resource planning/manufacturing execution system integration, cross-site optimization, model training, and archival storage. This hierarchy synthesizes proximity principles from edge computing [4], [7] with layered industrial architectures[1],[6].

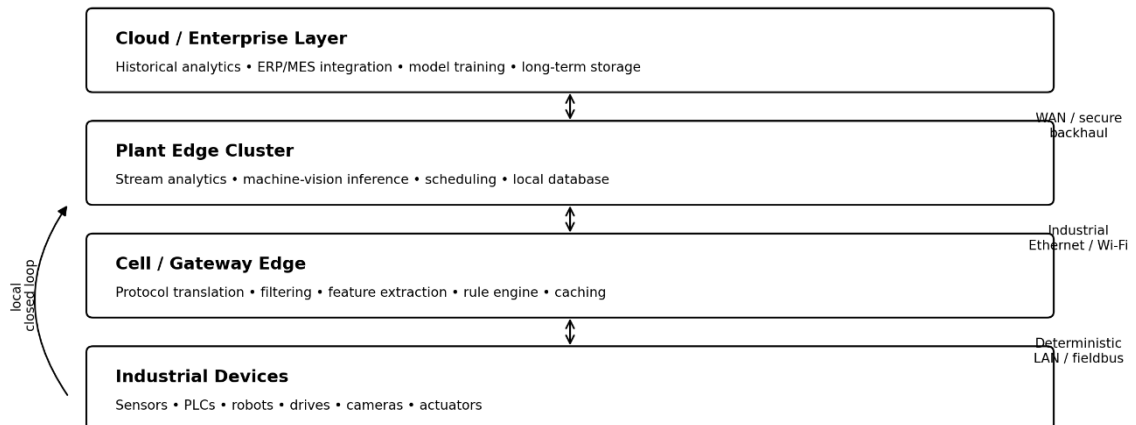


Figure 1. Proposed multi-tier edge computing architecture for smart industrial automation (author-developed synthesis based on [1], [4], [6], [7]).

Step 3: Apply deadline-aware task placement

For each job, the scheduler estimates the end-to-end latency for each eligible tier. The estimate is $T_{\text{total}} = T_{\text{access}} + T_{\text{queue}} + T_{\text{compute}} + T_{\text{return}}$. A tier is feasible when the estimated T_{total} is less than or equal to the job deadline L_j and when sufficient CPU, memory, and accelerator capacity are available. Among feasible tiers, the scheduler selects the nearest tier with adequate resources. Control-support jobs are restricted to Tier 2 or local Tier 1 logic; vision and monitoring jobs may execute at Tier 2 or Tier 3 depending on resource demand. Cloud execution is allowed only for non-critical jobs or as an explicitly controlled fallback. This is consistent with computation-offloading research

that treats communication and computing costs jointly [8], [11].

Step 4: Reduce data before WAN transmission

The architecture uses edge-side data reduction to minimize backhaul traffic. Raw high-frequency signals are filtered and summarized into features, events, or short windows before transmission. Images are retained locally unless a defect is detected or a sample is required for later model training. The gateway maintains a short persistent buffer so that measurements are not lost during temporary WAN outages. This approach follows the edge/fog rationale that network congestion and response time can be reduced when computation is moved closer to data sources [3]–[5].

Table 2. Deadline-aware workload placement policy used in the proposed architecture.

Workload class	Typical function	Primary tier	Deadline used	Cloud role
Control-support	Fast alarms, state estimation	Tier 1–2	40 ms	No real-time dependency
Machine vision	Inspection / defect inference	Tier 3	80 ms	Model training, archive
Condition monitoring	Feature extraction / anomaly scoring	Tier 2–3	100 ms	Fleet analytics, history

Step 5: Security, reliability, and interoperability controls

Edge nodes are treated as managed industrial infrastructure rather than unmanaged consumer devices. Communication between tiers uses authenticated channels, role-based service access, and signed software images. Device-facing gateways terminate heterogeneous protocols and expose normalized data objects upward, while the plant edge cluster separates operational-technology workloads from enterprise traffic through network segmentation. Local services continue operating if the cloud is unavailable, and cached events are synchronized after connectivity is restored. These controls address the heterogeneity and reliability concerns discussed in fog/edge research [9] and the flexible-network-management requirements identified for software-defined IIoT [6].

Step 6: Simulation-based evaluation procedure

A reproducible Monte Carlo simulation was used to compare three architectures over 5,000 synthetic jobs with a fixed random seed of 42. The workload mix was 50% monitoring, 30% vision, and 20% control-support. Payload sizes were sampled from log-normal distributions centered approximately at 20 KB, 250 KB, and 8 KB respectively. The cloud-only baseline sent every job through a simulated WAN and centralized compute queue. The gateway-plus-cloud baseline performed local preprocessing but still forwarded 60% of jobs for remote decisions. The proposed edge architecture executed control-support at the cell edge and most monitoring/vision workloads at the plant edge, with a 15% cloud fallback probability for non-critical processing. The simulation is an analytical evaluation of the architecture, not a claim of measurements from a deployed factory.

Table 3. Experimental comparison settings.

Parameter	Cloud-only	Gateway + cloud	Proposed edge
Jobs	5,000	5,000	5,000
WAN dependency	100%	60% after preprocessing	15% controlled fallback
Primary compute location	Remote cloud	Gateway + remote cloud	Cell / plant edge

Latency components	Access + WAN + queue + cloud	Local pre-process + optional WAN	Local/plant edge + optional fallback
Main objective	Elastic centralized compute	Reduce raw traffic	Meet deadlines near data source

Step 7: Define efficiency and acceptance criteria

The final methodological step defines how architectural efficiency is judged. Four metrics are used: mean end-to-end latency, 95th-percentile latency, deadline-success ratio, and average WAN data transmitted per job. Mean latency reflects typical responsiveness, while the 95th percentile captures delay variability that is more relevant to industrial service-level design. Deadline success is calculated as the number of jobs completed at or before their class deadline divided by the total number of jobs. WAN efficiency is measured as the mean number of kilobytes sent beyond the plant network. A design is accepted only if it improves deadline success without moving deterministic safety control away from the PLC/device tier. Thus, the evaluation rewards useful edge placement rather

than simply maximizing the amount of computation performed outside the cloud.

Results

Table 4 summarizes the simulation results. The cloud-only baseline achieved a mean response time of 114.4 ms and a 95th-percentile latency of 150.5 ms. Because the workload deadlines ranged from 40 to 100 ms, only 14.1% of cloud-only jobs completed within deadline. The gateway-plus-cloud design improved mean latency to 65.4 ms and deadline success to 60.6% by preprocessing data locally and avoiding WAN access for some jobs. The proposed multi-tier edge architecture achieved the lowest mean latency, 21.7 ms, and completed 95.2% of jobs within deadline. Its 95th-percentile latency was 79.9 ms, indicating that most tasks remained within the soft real-time envelope even when a small fraction used cloud fallback.

Table 4. Simulation results for 5,000 synthetic industrial jobs.

Metric	Cloud-only	Gateway + cloud	Proposed edge
Average latency (ms)	114.4	65.4	21.7
95th-percentile latency (ms)	150.5	119.9	79.9
Deadline success (%)	14.1	60.6	95.2
Average WAN data/job (KB)	89.0	15.0	2.7

Relative to cloud-only processing, the proposed architecture reduced average latency by approximately 81.0% and WAN data per job by approximately 97.0%. The deadline-success increase was especially important: the improvement from 14.1% to 95.2% shows that proximity is not merely a bandwidth optimization but a scheduling mechanism for time-sensitive industrial analytics. These findings are consistent with the core edge-computing argument that communication distance and remote queueing can dominate end-to-end delay for interactive or real-time services [4], [7], [8].

The remaining 4.8% deadline misses in the proposed model occurred primarily when jobs experienced cloud fallback or temporary edge queueing. This suggests two practical design priorities. First, edge capacity should be dimensioned using high-percentile rather than average demand. Second, fallback policy should be workload-aware: monitoring summaries can tolerate cloud diversion, whereas control-

support should degrade to a safe local rule rather than wait for a remote response. The results therefore support a hybrid edge-cloud model instead of an edge-only design. Cloud resources remain valuable for global optimization, model training, and long-term analytics, but they should not lie on the critical path of time-sensitive plant decisions.

The evaluation also illustrates the value of hierarchical preprocessing. Sending every raw observation to the cloud consumed an average of 89.0 KB per job. Gateway aggregation reduced this to 15.0 KB, while the proposed architecture reduced it further to 2.7 KB by retaining raw data locally and transmitting only selected events, summaries, or fallback requests. In a factory with continuous machine-vision and sensor streams, this reduction can lower backhaul requirements and isolate plant operation from WAN variability. Such locality aligns with the industrial communication trend toward integrating vertical information flows without sacrificing local automation performance [2].

From a deployment perspective, the results imply that edge efficiency depends on disciplined functional separation. The plant edge should not become a second uncontrolled data center. Services should be grouped by latency class, resource demand, and failure behavior, with capacity reserved for high-priority analytics. A local orchestration layer can monitor CPU, memory, accelerator utilization, and queue length, while an enterprise controller distributes application versions and policies during planned maintenance windows. This arrangement preserves the operational technology principle that loss of higher-level connectivity must not stop safe machine operation. It also allows edge nodes to be scaled incrementally by production cell rather than requiring a single large centralized platform.

Conclusion and Discussion

This paper presented a methodology for an efficient edge computing architecture tailored to smart industrial automation. The architecture separates deterministic device control, cell-level gateway functions, plant-level analytics, and cloud/enterprise services. Its key contribution is a deadline-aware placement policy that assigns each task to the nearest resource tier capable of satisfying latency and computational requirements. The design also uses local buffering, preprocessing, protocol normalization, and controlled cloud fallback to improve resilience and reduce WAN dependence. The simulation results indicate that the multi-tier design can substantially outperform cloud-only processing for latency-sensitive industrial workloads. Mean latency decreased from 114.4 ms to 21.7 ms, deadline success increased from 14.1% to 95.2%, and WAN traffic fell from 89.0 KB to 2.7 KB per job. These gains support the literature view that proximity, hierarchical computing, and local data reduction are central benefits of edge/fog systems [3]–[5], [7]. The architecture also remains compatible with Industry 4.0 principles because it preserves local automation responsibilities while adding cyber-level analytics and coordination [1], [6]. The study has limitations. The reported results are based on a controlled synthetic workload and simplified latency distributions rather than a hardware-in-the-loop factory deployment. Future validation should measure jitter, failure recovery, container start-up time, accelerator contention, and protocol-specific overhead on real PLCs, industrial Ethernet networks, gateways, and plant servers. Security evaluation should also include certificate management, software update integrity, and attack isolation. Nevertheless, the methodology provides a clear

2017-era reference design for deciding which industrial functions should remain local, which should move to the edge, and which can safely use centralized cloud resources.

References

- J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for Industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015. DOI: 10.1016/j.mfglet.2014.12.001.
- M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017. DOI: 10.1109/MIE.2017.2649104.
- F. Bonomi, R. Mito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16. DOI: 10.1145/2342509.2342513.
- W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. DOI: 10.1109/JIOT.2016.2579198.
- W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016. DOI: 10.1109/MC.2016.145.
- J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined Industrial Internet of Things in the context of Industry 4.0," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7373–7380, 2016. DOI: 10.1109/JSEN.2016.2565621.
- M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017. DOI: 10.1109/MC.2017.9.
- P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017. DOI: 10.1109/COMST.2017.2682318.
- S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop on Mobile Big Data*, 2015, pp. 37–42. DOI: 10.1145/2757384.2757397.
- M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: At the leading edge of mobile-cloud convergence," in *Proc. 6th Int. Conf. Mobile Computing, Applications and*

Services, 2014. DOI: 10.4108/icst.mobibase.2014.257757.

communication perspective," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, 2017. DOI: 10.1109/COMST.2017.2745201.

Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The