A Co-Designed KubeEdge–SDN Multi-Agent Framework for Secure and Predictable IIoT Manufacturing

Petro M. Tshakwanda*, Henok B. Tsegaye†, Ashok Karukutla‡, Raddad Almaayn§, Harsh Kumar‖, Michael Devetsikiotis**
*†‡§¶**Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, USA
‖R. B. Annis School of Engineering, University of Indianapolis, Indianapolis, USA
{*pmushidi, †htsegaye, ‡akarukutla, §raddadalmaayn, **mdevets}@unm.edu, {‖kumarh}@uindy.edu

***Abstract*—Industrial IoT (IIoT) networks increasingly rely on cloud–edge computing, but many deployments still struggle to meet tight latency, resilience, and security requirements in intelligent manufacturing. We present a co-designed KubeEdge–SDN–multi-agent framework that unifies Kubernetes-based orchestration, Ryu/OVS software-defined flow control, and Ray-based distributed agents into a single secure edge fabric. The framework enables low-latency anomaly detection, resilient failover, and SDN micro-segmentation for additive manufacturing workloads while keeping predictable performance at the Operational Technology (OT) edge. On a repeatable testbed, it sustains sub-100 ms anomaly-detection latency with high availability and keeps nearly all control events below 500 ms. Compared with EWMA, CUSUM, and Isolation-Forest baselines, our agents improve detection accuracy while respecting ISA-95 Level-2 timing constraints. The stack scales to 64 coordinated agents under adversarial load and provides a practical blueprint for secure, predictable Industry 4.0 deployments in additive manufacturing (AM).**

***Index Terms*—Edge Computing, KubeEdge, Software-Defined Networking (SDN), Multi-Agent Systems, Ray, IoT Manufacturing, Anomaly Detection, Kubernetes, Industrial IoT (IIoT).**

I. Introduction

The Industrial Internet of Things (IIoT) is reshaping manufacturing by linking sensors, actuators, and analytics to production processes [1], [2]. Cloud-centric designs offer scalability and central control, but backhaul delays and weak isolation between production cells make them unsuitable for control loops that need sub-second response and strong containment of faults or attacks [3], [4]. Edge computing reduces latency and backhaul load by moving computation closer to devices, yet many edge frameworks still (i) lack fine-grained network programmability, (ii) apply SDN without edge intelligence, or (iii) deploy agents without orchestration and security guarantees [5].

We propose an architecture that combines KubeEdge for edge orchestration, SDN via Ryu/Open vSwitch (OVS) for programmable connectivity, and Ray-based multi-agent intelligence for real-time analytics at the edge. The system is implemented on a repeatable testbed and evaluated for latency, resilience, scalability, and security under additive manufacturing workloads. It achieves low-latency inference, predictable tail behavior, fault-resilient failover, and effective micro-segmentation under adversarial conditions.

Section II reviews related work. Section III presents the architecture. Section IV explains the experimental setup. Section V reports results, and Sections VI–VII summarize findings and conclusions.

II. Related Work

Edge platforms such as EdgeX Foundry provide modular microservices but offer limited life-cycle and policy control compared to Kubernetes-native solutions [6]. FogBus2 improves fog/edge resource elasticity [7] but lacks container orchestration and fine-grained programmability, while Azure IoT Edge integrates with cloud analytics [8] yet remains proprietary. Our KubeEdge-based stack explicitly leverages Kubernetes scheduling, placement, and policy primitives while remaining open-source.

SDN has been integrated with edge/fog systems to improve traffic steering and resource control. FlexRAN demonstrates RAN programmability [9] but omits orchestration and distributed AI. SDN–IoT/fog surveys focus on resource allocation and traffic engineering [10] but are often decoupled from application workflows. Few IIoT systems bind SDN policy updates to application events with closed-loop guarantees. Here, the Ryu controller is embedded into the KubeEdge environment so that agent signals can drive SDN flow updates and their latency can be related to application performance.

Multi-agent paradigms enable distributed decision making for manufacturing control and anomaly response. Prior work on edge-resident agents [11] and agent-based IoT execution/testing [12] lacks (i) Kubernetes-grade life-cycle management and (ii) tight integration with programmable networking. We implement agents as Ray actors to obtain lightweight concurrency, flexible placement, and elastic scaling while retaining autonomy [13]. Security frameworks such as ENACT [14] and blockchain-based schemes [15] strengthen integrity and non-repudiation but often omit runtime network control.

Overall, existing work shows edge microservices, SDN programmability, or agents in isolation. Our contribution is a co-designed edge–SDN–agent stack for an IIoT AM cell that

couples agent intelligence to network policy in real time, is orchestrated with KubeEdge/Kubernetes, and unifies control, networking, and security.

III. Proposed System Architecture

A. Design Objectives

We target an additive manufacturing (AM) cell with multiple 3D printers, environmental sensors, and quality-control probes sharing constrained edge resources. The design focuses on:

- sub-second end-to-end latency with bounded tails (P_{95}) for control and quality-assurance loops;
- security and isolation via SDN micro-segmentation, least-privilege networking, and encrypted channels;
- elastic orchestration and observability using Kubernetes-native scheduling and metrics across application, network, and platform layers.

B. Architectural Overview and Components

Figure 1 shows three cooperating planes. **Data:** edge nodes ingest MQTT streams from printers and sensors; Ray agents perform online processing and anomaly inference on temperature, vibration, and process telemetry. **Control:** Kubernetes/KubeEdge manage life cycle and placement of services and agents; a Ryu controller programs OVS for flow steering and microsegmentation between printers, gateways, and monitoring services. **Telemetry:** Prometheus/Grafana collect metrics; NTP/PTP align timestamps so that end-to-end delay can be decomposed.

The framework consists of five roles. The **Master Node** hosts the Kubernetes control plane and Ray head. The **Cloud Node** runs KubeEdge CloudCore, the Ryu SDN controller, TLS termination, and time synchronization; it also exposes policy APIs for defining cell-level communication. **Edge Nodes** run KubeEdge EdgeCore, IoT applications with MQTT bridges, Ray workers, and OVS for per-flow isolation and traffic shaping; each typically serves one or more printers plus sensors, with pods pinned via labels and affinities. **Manufacturing Agents** are Ray actors that process per-printer streams, maintain local state, and emit intents (`isolate`, `prioritize`, `reroute`) compiled into SDN rules. The **Monitoring Stack** (Prometheus/Grafana) aggregates application, network, and platform metrics, including per-topic latencies, rule-install times, and node-level resource use.

KubeEdge–Kubernetes provides life-cycle management and placement [6]; SDN offers runtime network control; and Ray enables elastic edge autonomy. Fault tolerance combines Kubernetes deployments, readiness probes, and Ray actor reconstruction: if an edge pod or node fails, workloads are rescheduled on surviving `manufacturing-edge` nodes; agents reconnect and resume from local or remote state; MQTT's buffering helps avoid lost alerts.

C. Closed-Loop Operation and Security

The loop has six stages. **Ingestion**: sensors publish via MQTT, bridged by EdgeCore. **Inference**: Ray agents run Algorithm 2 on streams to detect deviations. **Intent emission**: agents issue intents such as `isolate(topic=X)` or `prioritize(flow=Y)`. **Policy synthesis**: the controller maps intents to SDN rules with priorities and timeouts, installed via Ryu on OVS. **Actuation**: OVS enforces microsegmentation and flow steering. **Feedback**: telemetry drives autoscaling, rescheduling, and policy updates.

Defense-in-depth includes TLS on control and telemetry channels, SDN microsegmentation enforcing least-privilege connectivity, and Kubernetes NetworkPolicies and RBAC. An optional L7 service mesh with mutual TLS can be enabled when per-request isolation is required. Section V quantifies policy-update latency and containment.

Algorithm 1 Edge-Aware Agent Orchestration

Require: K8s cluster K, edge nodes E, agent image I
Ensure: Deployed agents D with resource guarantees
1: Label eligible $e_j \in E$ as `mfg-edge`
2: For each agent a_i, set CPU/memory limits; enforce node affinity to `mfg-edge`
3: Init container checks `Ray` and `MQTT` with exponential backoff $\Delta_t = 5 \cdot 2^n$ s
4: Deploy a_i; register with Ray head; start telemetry exporters

Algorithm 2 Multi-Sensor Anomaly Detection

Require: Sensor stream S, thresholds θ
Ensure: Alerts A
1: **for** reading $s_i = (v_i, t_i, \tau_i) \in S$ **do**
2: **if** $\tau_i \in \theta$ **then**
3: $A_i \leftarrow \mathbb{1}[\, v_i \notin [\theta^{\min}_{\tau_i}, \theta^{\max}_{\tau_i}]\,]$
4: **else**
5: Update μ, σ; $A_i \leftarrow \mathbb{1}[\, |v_i - \mu| > 3\sigma \,]$
6: **end if**
7: **if** $A_i = 1$ **then**
8: Publish $\langle \text{agent}, \tau_i, v_i, t_i \rangle$ and emit SDN intent if needed
9: **end if**
10: **end for**

D. Scheduling and Detection Logic

KubeEdge extends Kubernetes scheduling to heterogeneous edge nodes using labels and (anti-)affinity. Printers and sensors attach to nodes labeled `manufacturing-edge`, and agents are scheduled with node affinity to keep inference close to the data. Algorithm 1 sketches the orchestration path, including node qualification, resource bounds, and a fault-tolerant init phase. Agents combine domain thresholds for known sensor types with adaptive bounds $\mu \pm 3\sigma$ over a sliding window for others (Algorithm 2). Alerts carry context and may trigger SDN intents, printer pausing, or job rescheduling.

E. Timing and Metrics

At measurement points A–D (Figure 2), each message i carries timestamps $t_X^{(i)}$ for $X \in \{A, B, C, D\}$. Latencies are $L_{X \to Y}^{(i)} = t_Y^{(i)} - t_X^{(i)}$, with $L_{A \to C}$ capturing anomaly-detection latency and $L_{B \to D}$ export latency. We report mean, P_{50}, P_{95}, and 95% confidence intervals.

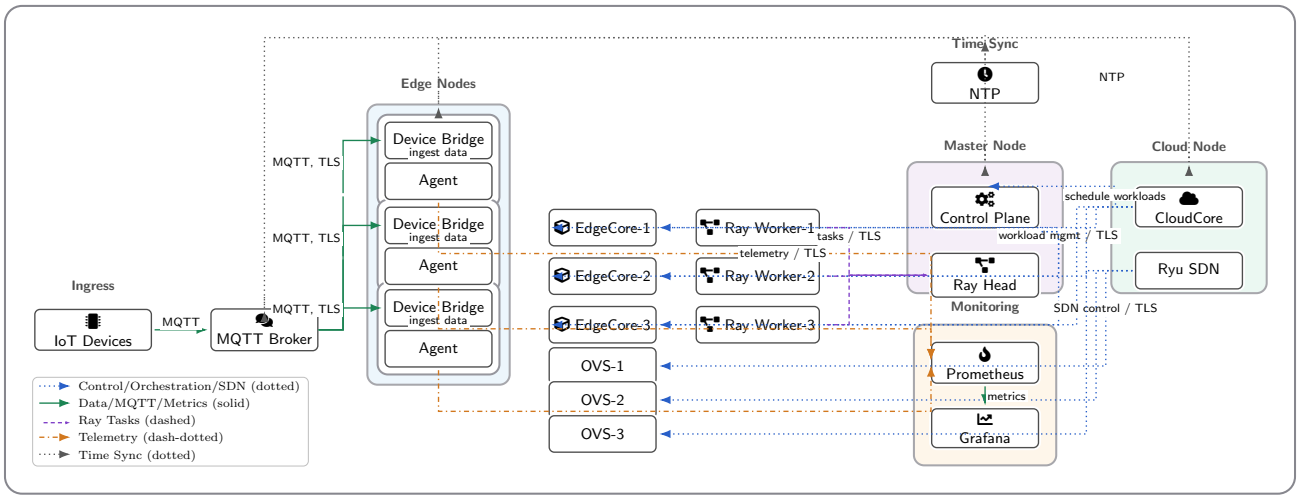

Fig. 1: Architecture with data, control, and telemetry planes across Edge, Master/Monitoring, and Cloud for an additive manufacturing cell.

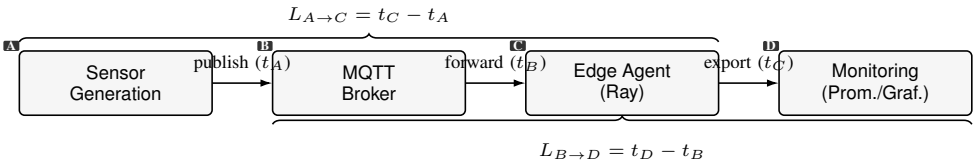

Fig. 2: Measurement pipeline (points A–D) for the AM cell.

IV. Evaluation Methodology

A. Testbed and Workloads

One *master* and one *cloud* node host the Kubernetes control plane, KubeEdge CloudCore, and Ryu controller; multiple *edge* nodes run KubeEdge EdgeCore, MQTT bridges, Ray workers, and OVS. Edge nodes emulate AM gateways attached to printers and sensors. NTP/PTP and TLS-secured channels ensure consistent and secure operation. IoT sensor streams arrive via MQTT with payloads of 128 B–8 KB and configurable burst profiles, representing temperature, vibration, and status messages from 3D printers. Anomaly detection is evaluated on labeled synthetic traces augmented with manufacturing-style telemetry, including normal print phases and injected faults (overheating, misalignment, abnormal vibration). Separate burst experiments stress the pipeline and emulate intensive production or failure storms.

B. Metrics, Ablations, and Baselines

Instrumentation at points A–D (Figure 2) yields $L_{A\to C}$ and $L_{B\to D}$, throughput (msg/s), loss, CPU (millicores), memory (MiB), and SDN rule-install latency. Each experiment is repeated with randomized seeds and independent deployments; we report mean, P_{50}, P_{95}, and 95% confidence intervals.

To attribute gains, we evaluate: *No-SDN* (Ryu/OVS disabled; static connectivity) and *No-Agents* (centralized analytics; no edge Ray workers). The detector is compared against EWMA, CUSUM, and Isolation Forest, reporting ROC/PR curves, AUC_{ROC}/AUC_{PR}, and F1; thresholds are chosen to maximize F1. We also scale agents/pods, sweep payload sizes and burst rates, and inject scripted link, switch, controller, and node faults to derive recovery timelines and availability. The threat model covers man-in-the-middle, rogue device onboarding, and lateral movement, and ISA-95 L2 conformance is assessed via latency/jitter and workload representativeness [16]. Manifests, policies, and Prometheus queries are versioned to support reproducibility.

V. Results

A. Latency, Throughput, and Availability

Across 600 s runs with four agents on two pods, end-to-end anomaly-detection latency averages **64.07 ms** with P_{95}=64.8 **ms**. Under *No-Agents*, mean latency increases by **39%**; under *No-SDN*, failover after perturbations slows by **7.8** × (see also Table V), indicating that both edge intelligence and SDN contribute to predictable response.

Figure 4 shows message accumulation with 95% CI ribbons; rates are stable and match

$$N(t) \approx 1.21\,t + 3.47, \tag{1}$$

with observed availability of **99.95%**. Overall, **99% of events complete** < 500 **ms**, meeting the AM cell SLO (Table I).

B. Resource Utilization and SDN Overhead

CPU/memory results (Figure 5 and Table III) show lightweight application pods (∼9.5 m CPU, ∼95 MiB), with system services and monitoring dominating overhead, which is

TABLE I: Availability and SLO conformance.

Metric	Definition	Observed
Availability	$1 - \frac{\text{downtime}}{\text{obs. window}}$	**99.95%**
SLO (sub-500 ms)	$\Pr[L_{A\to D} < 500\ \text{ms}]$	**99%**
Throughput (steady)	$\bar{\lambda}$ over 2000 s	**1.21 msg/s**

TABLE II: Anomaly-detection accuracy (F1).

Method	F1
Ours (edge Ray)	**0.92**
EWMA	0.74
CUSUM	0.79
Isolation Forest	0.85

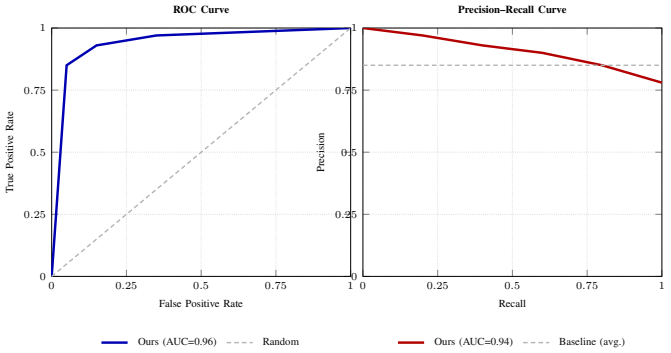

Fig. 3: ROC and PR performance of the multi-agent anomaly detector.

useful for sizing industrial edge gateways. We do not directly instrument power or temperature, but the low CPU duty cycle and modest memory footprint are compatible with fanless edge-class hardware.

Flow-rule install latency $L{=}t_{\text{complete}}{-}t_{\text{initiate}}$ over 600 s has mean **11.39 ms** and $P_{95}{=}14$ **ms** (Table IV), supporting closed-loop responsiveness when agent intents reconfigure paths or isolate printers.

TABLE IV: Flow-rule install latency (600 s window).

Metric	Value (ms)	Std. Dev.
Min	9.00	–
Median	11.00	–
Mean	11.39	1.40
P_{95}	14.00	–
Max	15.00	–

C. Ablation, Security, and Scalability

Table V summarizes ablation results. Disabling SDN increases recovery time from 3.2 s to 25.6 s (7.8×), while removing Ray agents increases mean latency to 89 ms (39% higher).

TABLE III: Resource utilization summary (all nodes).

Component	CPU (m)	Memory (MiB)	Pod Count
Application Pods	9.5	95	2
System Services	38.2	143	23
Monitoring	19.8	214	7
Network	32.4	89	6

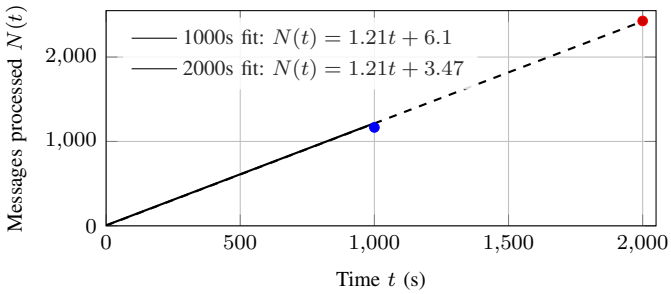

Fig. 4: Message accumulation and linear fits with 95% CI ribbons.

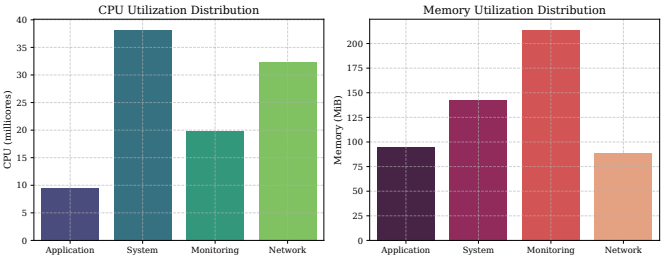

Fig. 5: CPU/Memory utilization distribution across node components.

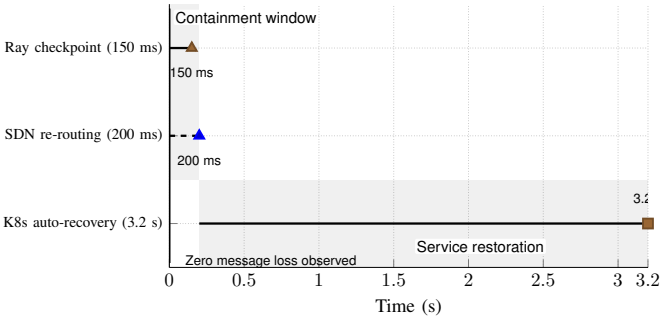

Fig. 6: Recovery timeline: 150–200 ms containment and 3.2 s auto-recovery.

Qualitatively (Table VI), both ablations worsen latency and tails; SDN also hurts loss and recovery.

Under adversarial traffic, SDN micro-segmentation contains lateral flows, and policy updates are applied within Table IV bounds. TLS protects control and telemetry channels, and an optional service mesh provides mutual TLS at Layer 7. We scale to **64** distributed agents with near-linear throughput growth (consistent with Figure 4) and no observed message loss (Figure 6). The evaluated AM workload satisfies ISA-95 Level 2 requirements [16]: 99% of events $<$ 500 ms, deterministic segmentation between logical cells, and repeatable orchestration actions.

TABLE V: Ablation results with means, tails, and 95% CIs (deltas in parentheses).

Variant	Mean L [ms]	P95 [ms]	CI95 [ms]	R [s]
Full stack	64.07	64.80	±0.37	3.20
No-Agents	89.00 (+39%)	—	±0.50	3.20
No-SDN	66.00 (+3%)	—	±0.45	25.60

TABLE VI: Ablation effects (qualitative). Legend: ↑ worse, ↔ similar.

Variant	Latency	Tails	Loss	Recovery
No-SDN (static flows)	↑	↑	↑	↑
No-Agents (centralized)	↑	↑	↔	↔
Full stack (ours)	baseline	baseline	0 (obs.)	bounded

VI. Discussion and Key Findings

The co-designed cloud–edge–SDN–agent stack keeps latency-critical control at the edge while using centralized coordination for policy and life-cycle management. Anomalies at printers are handled within tens of milliseconds, and placement and traffic policies remain consistent. The hybrid detector outperforms EWMA, CUSUM, and Isolation Forest (F1$=0.92$ vs. 0.74–0.85; $\mathrm{AUC}_{\mathrm{ROC}}=0.96$, $\mathrm{AUC}_{\mathrm{PR}}=0.94$) (Table II and Figure 3), while SDN control-plane latency remains low (mean 11.39 ms, $P_{95}=14$ ms) and micro-segmentation contains lateral flows. Ablations confirm that edge agents and SDN are both important for resilience: disabling SDN slows failover by $7.8\times$, and removing agents increases latency by 39% (as shown in Table V). The system scales to 64 agents with stable throughput and no observed loss (Figure 6).

VII. Conclusion

We presented a co-designed cloud–edge–SDN multi-agent architecture that unifies KubeEdge orchestration, Ryu/OVS programmability, and Ray-based analytics for an IIoT additive manufacturing cell. On a repeatable testbed, the system delivers sub-100 ms end-to-end detection ($\bar{L}=64.07$ ms, $P_{95}=64.8$ ms), stable throughput (1.21 msg/s) with 99.95% availability, tight SDN rule-install times (mean 11.39 ms), high detection accuracy (F1$=0.92$ vs. 0.74–0.85), and scalability to 64 agents while meeting ISA-95 Level 2 timing. Ablations show that programmable networking and edge intelligence are both necessary for predictable, resilient operation in the AM cell. The architecture offers a practical blueprint for secure, predictable Industry 4.0 deployments that combine edge orchestration, SDN, and multi-agent intelligence. Future work includes cross-site federated learning, live agent migration, hardware-in-the-loop validation, explicit energy and thermal measurements on edge hardware, and time-sensitive networking to further tighten latency tails.

Acknowledgment

This work was supported by the US National Science Foundation under the Award OIA-2417062 to the DREAM Research Center and the UNM IoT and Intelligent Systems Innovation Lab (I2 Lab).

References

[1] V. V. Baligodugula, A. Ghimire, and F. Amsaad, "An overview of secure network segmentation in connected iiot environments," *Computing&AI Connect*, vol. 1, no. 1, pp. 1–10, 2024.

[2] Y. Qiao, A. S. Hafid, N. Agoulmine, A. Karamoozian, L. Tamazirt, and B. Lee, *Edge Computing and Distributed Intelligence*. Cham: Springer International Publishing, 2024, pp. 129–145. [Online]. Available: https://doi.org/10.1007/978-3-031-39650-2_7

[3] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, "Edge computing: current trends, research challenges and future directions," *Computing*, vol. 103, no. 5, pp. 993–1023, 2021.

[4] A. A. Mirani, G. Velasco-Hernandez, A. Awasthi, and J. Walsh, "Key challenges and emerging technologies in industrial iot architectures: A review," *Sensors*, vol. 22, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5836

[5] C. Savaglio, P. Mazzei, and G. Fortino, "Edge intelligence for industrial iot: Opportunities and limitations," *Procedia Computer Science*, vol. 232, pp. 397–405, 2024, 5th International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050924000395

[6] S. Taherizadeh, V. Stankovski, and J.-H. Cho, "Dynamic multi-level auto-scaling rules for containerized applications," *The Computer Journal*, vol. 62, no. 2, pp. 174–197, 2019.

[7] Q. Deng, M. Goudarzi, and R. Buyya, "Fogbus2: a lightweight and distributed container-based framework for integration of iot-enabled systems with edge and cloud computing," in *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments*, ser. BiDEDE '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3460866.3461768

[8] M. N. Jamil, O. Schelén, A. Afif Monrat, and K. Andersson, "Enabling industrial internet of things by leveraging distributed edge-to-cloud computing: Challenges and opportunities," *IEEE Access*, vol. 12, pp. 127 294–127 308, 2024.

[9] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "Flexran: A software-defined ran platform," in *Proceedings of the 23rd annual international conference on mobile computing and networking*, 2017, pp. 465–467.

[10] S. Shafiq, M. S. Rahman, S. A. Shaon, I. Mahmud, and A. S. Hosen, "A review on software-defined networking for internet of things inclusive of distributed computing, blockchain, and mobile network technology: Basics, trends, challenges, and future research potentials," *International Journal of Distributed Sensor Networks*, vol. 2024, no. 1, p. 9006405, 2024.

[11] P. Papcun, E. Kajati, D. Cupkova, J. Mocnej, M. Miskuf, and I. Zolotova, "Edge-enabled iot gateway criteria selection and evaluation," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 13, p. e5219, 2020.

[12] J. P. Dias, F. Couto, A. C. Paiva, and H. S. Ferreira, "A brief overview of existing tools for testing the internet-of-things," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2018, pp. 104–109.

[13] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging ai applications," 2018. [Online]. Available: https://arxiv.org/abs/1712.05889

[14] N. Ferry, A. Solberg, H. Song, S. Lavirotte, J.-Y. Tigli, T. Winter, V. Muntés-Mulero, A. Metzger, E. Rios Velasco, and A. Castelruiz Aguirre, "Enact: Development, operation, and quality assurance of trustworthy smart iot systems," in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, J.-M. Bruel, M. Mazzara, and B. Meyer, Eds. Cham: Springer International Publishing, 2019, pp. 112–127.

[15] X. Feng, J. Wu, Y. Wu, J. Li, and W. Yang, "Blockchain and digital twin empowered trustworthy self-healing for edge-ai enabled industrial internet of things," *Information Sciences*, vol. 642, p. 119169, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025523007545

[16] International Society of Automation (ISA), "ANSI/ISA-95.00.01-2025 (IEC 62264-1 Mod): Enterprise-Control System Integration – Part 1: Models and Terminology," https://www.isa.org/products/ansi-isa-95-00-01-2025-iec-62264-1-mod-enterprise, ISA, Research Triangle Park, NC, 2025.