Leveraging Edge and Fog Resources While Complying with EU's GDPR

Matilde Silva, Pedro C. Diniz[a] and Gil Gonçalves[b]
SYSTEC, Faculdade de Engenharia, Universidade do Porto, 4200-465 Porto, Portugal

Keywords: Edge Computing, Fog Computing, Industrial Internet of Things (IIoT), Industry 4.0, Fault Tolerance, IEC 61499, Real-Time Data Processing.

Abstract: In Industry 4.0 environments, video-based monitoring systems must now reconcile performance demands with the privacy mandates of the European Union's (EU) General Data Protection Regulation (GDPR). This paper presents a fault-tolerant edge/fog architecture designed to anonymize visual data at the point of capture, minimizing personal data exposure while maintaining low-latency analytics. Built on the IEC 61499 standard, the system uses DINASORE to run Function Blocks directly on edge devices, and T-Sync as an orchestrator that dynamically reallocates tasks in response to topology changes. Empirical evaluations demonstrate that the architecture reliably recovers from node loss and stays within resource limits even on modest hardware. Despite bottlenecks under heavy vision workloads, the results show the viability of deploying GDPR-compliant IIoT pipelines without offloading sensitive data to the cloud.

1 INTRODUCTION

Internet of Things is an ubiquitous tool in the realm of industry and is identified as the Industrial Internet of Things (IIoT), with broad and diverse application areas in sectors such as agriculture, environmental monitoring, security surveillance, and others (Xu et al., 2014).

The vast amounts of data generated by IIoT sensors can be used to analyse, and even optimize, the system's performance, *i.e.* by using the information to train Artificial Intelligence and Machine Learning algorithms (Murugesan, 2016).

In the long run, IIoT data analytics can improve performance and reduce spending costs, both of which greatly benefit a company (Jeschke et al., 2017). As such, industrial manufacturers are increasingly interested in research that helps in efficient resource usage.

Although these efficiency gains are attractive, the same sensor networks now gather large volumes of personal data, especially video streams that capture employees and visitors on the industrial floor. In the EU, such data are protected by the General Data Protection Regulation (European Parliament and Council of the European Union, 2025), which forces data controllers to minimize the amount of personally identifiable information they collect.

A concrete example of this comes from Bosch Ovar. Warehouse cameras that track stock movements inevitably record workers passing through the scene. To avoid the administrative burden of obtaining individual consent for every employee, it is required that raw frames be anonymized before any footage is stored or forwarded upstream.

Current architectures that rely on edge and fog computing are not equipped to handle large amounts of video data and to be part of a system that is highly volatile and needs to tolerate and accommodate spontaneous changes, without needing manual re-deployment.

To overcome these limitations, we propose an edge and fog architecture in which lightweight IEC 61499 Function Blocks run on DINASORE-enabled edge devices, while a T-Sync orchestrator automatically redeploys changes of workload as the physical topology evolves.

This article thus makes the following contributions:

- It describes the development of a fault-tolerant Edges/Fog IEC 61499-based system, centred around function blocks and modular software components.
- It presents experimental results that show the effectiveness of a dynamic task orchestrator in response to topology changes while maintaining ad-

[a] https://orcid.org/0000-0003-3131-9367
[b] https://orcid.org/0000-0001-7757-7308

Silva, M., Diniz, P. C. and Gonçalves, G.
Leveraging Edge and Fog Resources While Complying with EU's GDPR.
DOI: 10.5220/0013781300003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 518-525
ISBN: 978-989-758-770-2; ISSN: 2184-2809

equate Quality of Service (QoS) regarding a video stream and the anonymization of video images.

- Its shows that it is possible to build a fault-tolerant Edges/Fog image acquisition system while complying with the EU's stringent GDPR requirements.

The pervasiveness of IIoT and in particular the use of video processing capabilities coupled with reliable video capture analysis for security and enhanced performance in factory settings is a growing need. To be effective, these systems need to be both reliable and support the EU's GDPR. This article shows that it is possible to develop Edge/Fog systems that can meet all functional and quality requirements, with an architecture that is both flexible and reliable.

The remainder of this paper is structured as follows. In the next section 2, we describe the methodology of the literature review, highlighting related work, as well as technologies. Section 3 describes the IEC 61499 edge/fog architecture and its implementation. In Section 4, we present empirical results from a fault, stress and topology evaluation methodology and we conclude in Section 5 with key takeaways and avenues for future research.

2 RELATED WORK

2.1 Systematic Literature Review

The systematic review began with defining core concepts such as Industrial IoT, edge- and fog-based computing, and decentralised orchestration. Using these terms, searches were run across four databases: (1) IEEE Xplore, (2) ACM Digital Library, (3) Scopus and (4) ScienceDirect.

A series of filters were then applied, *i.e.* English language, peer-reviewed status, and a publication window from 2019 onwards.

2.2 State of the Art

The current body of work shows a shift from cloud-only pipelines toward edge- and fog-centred IIoT architectures (Bonomi and Milito, 2012; Mahmud et al., 2018). Researchers point to network-saturation risks, the heterogeneity of devices and the need for millisecond-scale feedback as key drivers for this change (Murugesan, 2016). Microservice designs have also become popular because they break monoliths into independently deployable bits, yet they also introduce heavier inter-service coordination and higher management overhead once dozens of cameras or sensors join the network (Torvekar and Game, 2019; Vural et al., 2017; Dinh-Tuan et al., 2019). At the same time, real-time constraints and the push for event-driven behaviour keep solutions drifting away from traditional Service Oriented Architectures (SOA) stacks toward lighter, decentralised runtimes.

Despite that progress, two blind spots remain. First, privacy is still treated almost exclusively as a transport problem, while the GDPR principle of data minimisation is not taken into account (Liu et al., 2020). Second, most orchestrators assume a static inventory of edge nodes, which forces engineers to step in and re-deploy services by hand when necessary (Hu et al., 2024; Etemadi et al., 2020). These manual touchpoints undo the resilience and scalability promised by edge computing and leave factories with a more rigid infrastructure. Bridging these gaps calls for an architecture that anonymises data at the point of capture and re-allocates workloads automatically as the topology shifts, without sacrificing the deterministic behaviour demanded by industrial control.

2.3 Enabling Technologies

Among the frameworks already recognised by the community, the IEC 61499 standard stands out as the most mature option for event-driven automation. Its Function-Block model cleanly separates behaviour from deployment, letting the same logic migrate between devices without altering timing semantics; a property that aligns well with the low-latency, reconfigurable needs surfaced in the previous section (Vyatkin, 2016). Because the standard also mandates explicit data-and-event interfaces, it naturally supports fine-grained placement decisions at run time while preserving the determinism expected on a safety-critical line.

Complementing the standard, the Eclipse 4Diac tool provides a graphical IDE for composing Function Block (FB) networks(Eclipse Foundation, 2024). Together, IEC 61499 and 4Diac form an off-the-shelf foundation on which more adaptive, privacy-aware architectures can be built.

Extending this foundation, DINASORE offers a lightweight Python runtime that executes IEC 61499 networks directly on the edge devices, allowing access to modern Python libraries without breaking the event-driven semantics of the standard (Pereira et al., 2020). Each FB becomes a self-contained Python module, fully customizable and editable.

Dynamic placement of those blocks is handled by T-Sync, a minimalist orchestrator written in Python

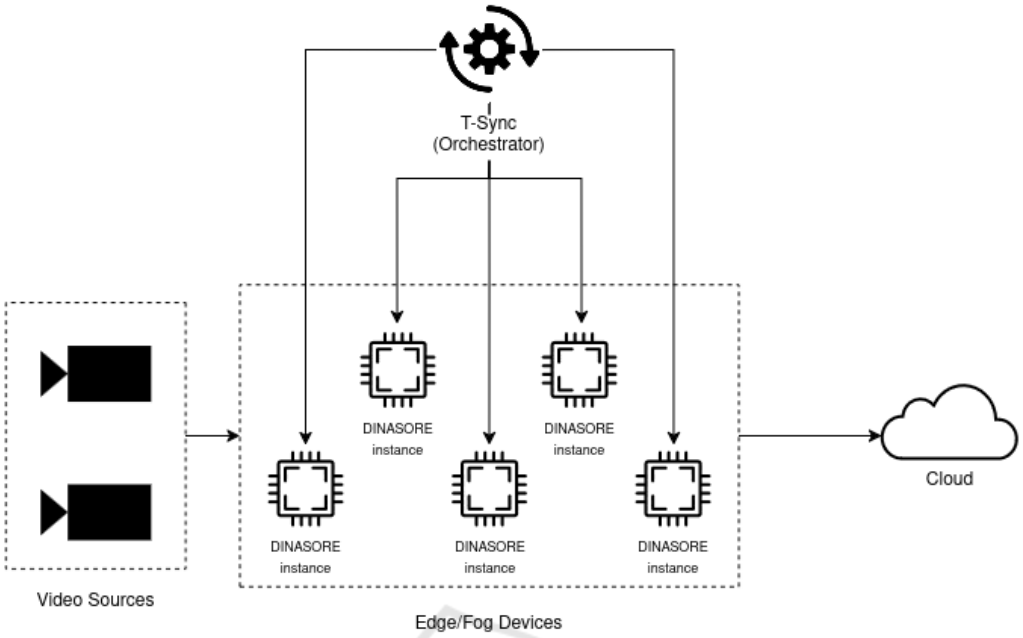

Figure 1: High-level diagram of system architecture.

(Pereira and Gonçalves, 2025). It integrates with DINASORE to facilitate dynamic reconfiguration of Cyber Physical Production Systems (CPPS) by implementing a resource optimization algorithm that performs optimal FB assignment across available devices. T-Sync continuously monitors system performance through metrics collection, and uses the TREAO genetic algorithm to determine optimal task placement, and automatically deploys new configurations when improvements are identified (Pereira et al., 2024).

Together with DINASORE, T-Sync addresses the challenge of efficient resource utilization and system reconfiguration by providing automated, optimized distribution of computational tasks across networked devices while maintaining compliance with industrial standards.

3 IMPLEMENTATION

Continuing with the previous Bosch Ovar example, we assume a scenario in which cameras capture images of the factory floor, and edge devices that belong to the system perform the pre-processing to remove sensitive, personal information.

To mitigate the raised issues we propose an architecture that is based on the IEC 61499 industrial standard. This standard focuses on function blocks and their importance in outlining industrial processes, while also being centred around an event-driven execution model.

The proposed architecture would be triggered by events, *e.g.* the detection of a person, the addition of a new processing device or camera to the system, or removal/failure of other nodes and cameras. The infrastructure is assumed to be observable, in other words, when a change in topology occurs, there will be an event notifying such behaviour. As a consequence, the proposed architecture will have an orchestrator logic, performed by T-Sync, which will receive the updates and delegate tasks accordingly.

3.1 Architecture Design

The initial requirements for the design on the architecture is that it be decentralized, adaptable, event-driven, fault tolerant and scalable.

Figure 1 illustrates the high-level diagram for the proposed architecture. In it, there are four main components, *i.e.* the video sources, the edge/fog devices, the orchestrator and the cloud. The video sources generate and transmit the images via Real-Time Streaming Protocol (RTSP).

The cameras communicate directly with the edge/fog devices and, in turn, these devices communicate with the cloud; however, there is no communication between each device within the edge and fog layer. Instead, the orchestrator tells which device what to do. This design choice avoids the complex-

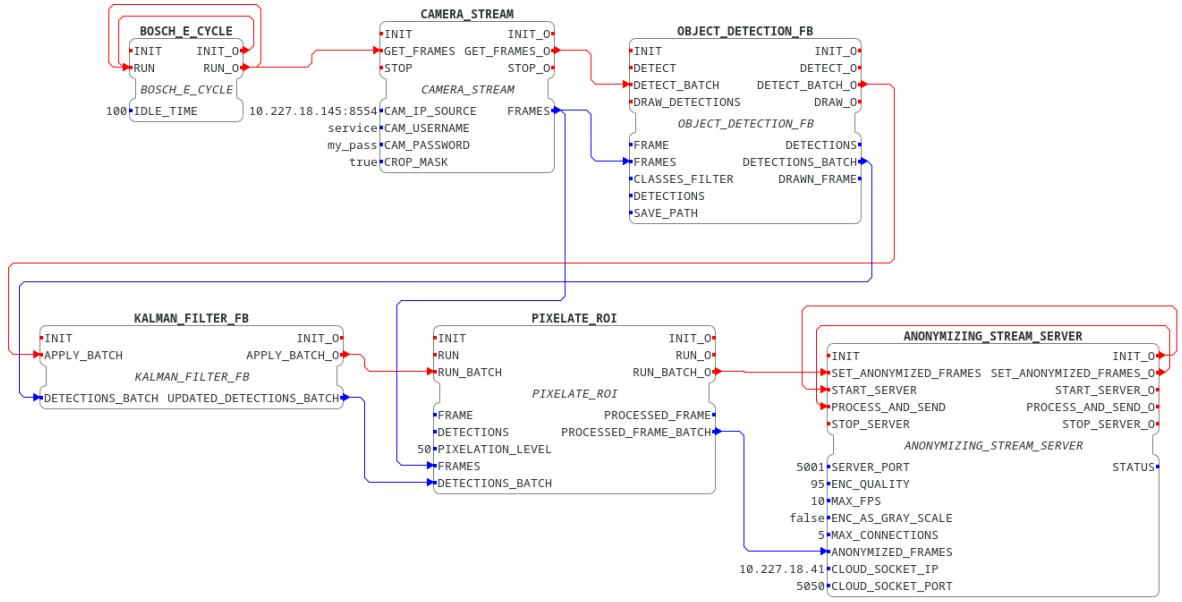

Figure 2: Complete Function Block Anonymization Pipeline.

ity of peer-to-peer communication and simplifies fault detection and task reassignment.

Furthermore, it is assumed that the orchestrator is aware of any topology changes within the network. Since the primary focus of this research is on the architectural design, the mechanisms for network discovery and monitoring are considered out of scope.

3.2 Development

The architecture centres on the IEC 61499 standard, using the Function Block (FB) as basis. As a result, the development starts by creating the function blocks most relevant to the problem at hand.

In order to anonymize images, for each frame, the following sequence must occur: (1) capture the video frame, (2) identify and locate any humans, (3) apply a filter for detection stability, (4) pixelate regions of interest and, lastly, (5) transmit the frame through a server socket accessible to the cloud. With this process limited to the edge and fog boundaries, it's guaranteed that personal information never reaches centralized storages.

For each stage of the sequence, a function block was created, in addition to an E_CYCLE function block, which polls the cameras for frames every N seconds. The creation of function blocks means the creation of a XML file with the specified input/output variables and events, and a Python file in containing the logic to be executed for each event. This format is required because of integration with 4Diac and DINASORE.

With each function block created, the whole process must be plotted out in 4Diac, attaching each function block to each other. This is illustrated in Figure 2.

Once the anonymisation pipeline is modelled in 4Diac, the XML description is handed off to T-Sync, which keeps a static record of the available processing devices, the `dev_specs.json` file. On start-up, T-Sync parses this file and, together with the 4Diac application XML, produces a single placement plan, pushing it to the processing devices which should be running DINASORE. The task allocation is handled by TREAO, as previously mentioned.

From that moment onward, the orchestrator would redeploy the task configuration if a timeout was reached (an option offered by T-Sync). If a processing device failed or a new camera was installed, there had to be a manual intervention to change the `dev_specs.json`, followed by a T-Sync restart to bring the change into effect.

As the intent is to make this architecture adaptive, this was a flaw that needed to be addressed. To achieve this, a Python script was added to the execution of T-Sync. The Python script receives changes from the network, which are always assumed to be truthful and correct, and changes the `dev_specs.json` file accordingly, updating it during the T-Sync runtime. In addition, a flag is raised whenever this file is altered, then when T-Sync sees the flag change, it re-runs the task allocation algorithm, with the new device details, and re-deploys the new tasks.

Taken together, these design and implementation

steps tackle the original aims of the project. Processing remains close to the cameras, and the system reorganises itself whenever devices appear, fail, or are upgraded.

4 EVALUATION

To assess the architecture's robustness and performance, we evaluated the following metrics, as defined more precisely below: (1) MTTR (Mean Time To Recover), (2) Data Loss Rate, (3) End-to-end latency, (4) Frame processing rate, (5) CPU usage and (6) Memory usage. This set of criteria will give us a robust insight as to the performance of the developed architecture in the face of different challenges and situations.

4.1 Experimental Setup

For this evaluation a testbed with the following computing platforms and cameras was setup, as shown on Figure 3, thus including a total of three processing devices and two cameras.

1. Raspberry Pi 4 Model B (2 GB RAM)[1] with Raspberry Pi Camera Module 2[2]
2. Raspberry Pi 5 (16 GB RAM)[3] with Raspberry Pi AI HAT+ (26 TOPS)[4]
3. Raspberry Pi 5 (16 GB RAM)[3] with Raspberry Pi AI HAT+ (13 TOPS)[4] and Raspberry Pi AI Camera[5]

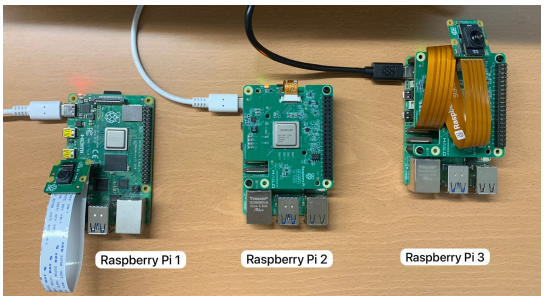

Figure 3: Raspberry Pi evaluation testbed.

[1]Raspberry Pi 4 Model B product page
[2]Raspberry Pi Camera Module 2 product page
[3]Raspberry Pi 5 product page
[4]Raspberry Pi AI HAT+ product page
[5]Raspberry Pi AI Camera product page

4.2 Empirical Evaluation

To evaluate the performance of the proposed approach 4 different types of experiments were conducted. All experiments share the same hardware pipeline, yet the relevance of each metric differs with the goal of the test.

4.2.1 Baseline Performance Test

For this test we conducted a controlled experiment with no disruptions, which aims to capture the system's default behaviour under standard operation conditions. This will serve as the benchmark for later comparisons. The experiment used the 3 Raspberry Pis and 2 cameras (each at 10 FPS) all throughout, operating uninterrupted for four minutes.

Four metrics stand out for this test: (1) **End-to-end latency** provides the reference QoS level under typical load; (2) **Frame processing rate** determines the rate at which the system operates; (3) **Data loss rate** gives insight as to the baseline losses for further comparison; (4) **CPU/Memory usage** captures the steady-state resource footprint, used to determine computing overloads.

Table 1 details the acquired metrics for the baseline test. The measured end-to-end latency of **3.4 seconds**, with OBJECT_DETECTION_FB dominates the computation budget, requiring on average 3.124 ± 2.629 seconds to analyse one frame. The system processes only 0.62 frames per second while both cameras supply the video feed at 10 FPS. Accordingly, about 94% of frames are discarded. The bottleneck is unequivocally caused by `OBJECT_DETECTION_FB`, which accounts for almost all the computational budget. The standard deviation highlights the discrepancy between devices with hardware accelerators (Raspberry Pi 2 and 3) and those without (Raspberry Pi 1).

Table 1: Average captured metrics for baseline test across all devices.

Metric	Value
Frames captured	2 728
Frames streamed	156
Frame processing rate (s^{-1})	0.62
Data loss rate (%)	94.3
End-to-end latency $\pm\sigma$ (s)	3.42 ± 0.30

Despite the backlog, processor utilisation remained below 0.2% on all devices, an indication that the workload is overwhelmingly I/O-bounded rather than CPU-bounded.

As a whole, the baseline experiments revealed a system whose latency is constrained by the processing

Table 2: Average CPU and memory usage per device for baseline test.

Device	CPU (%)	Memory (MB)
Raspberry Pi 1	0.105 %	114
Raspberry Pi 2	0.047 %	72
Raspberry Pi 3	0.020 %	147

rate of the miscellaneous devices, even when some are equipped with hardware accelerators.

4.2.2 Failure Tests

Regarding the failure tests, the setup is identical to the set up used for the baseline test. In this case, the system starts with the 3 Raspberry Pis and 2 cameras, after which 2 devices and 1 camera were disconnected from the system, leaving a single Raspberry Pi and camera. In this case, Raspberry Pi #2 and the camera attached to Raspberry Pi #3 were the sole survivors.

For this test, the relevant metrics are: (1) **MTTR** which measures the speed at which the system can recover to gauge behaviour in the face of failures; (2) **End-to-end latency** which analyzes latency under failures to determine possible behaviour changes; (3) **Data loss rate** to understand the amount of data lost, compared to the baseline, during a failure; (4) **CPU/Memory usage** to verify if during failures devices become overwhelmed.

For analysis, the logs were partitioned into two segments, pre-fault and post-recovery. Each stage lasted about a minute, not including recovery time. The results are shown on table 3. The first deployment starts at 0.00 seconds, the topology-change event is recorded at 56.82 seconds, and normal service resumes with a second deployment at 99.86 seconds. The difference between the latter two timestamps is **43 seconds**, which is the MTTR for this run.

Table 3: System performance before and after recovery during failure test.

	Pre-fault	Post-recovery	Δ
Duration (s)	56.82	55.85	-
Frames captured	648	448	-30.9%
Frames streamed	46	77	+67.4%
Processing rate	0.81	1.38	+70.3%
Data-loss rate (%)	92.9	82.8	-10.1%
End-to-end latency $\pm\sigma$ (s)	5.27 ± 2.22	0.80 ± 0.07	-84.8%

Table 4: Average CPU and memory usage during failure test.

	CPU (%)		Memory (MB)	
Device	Pre	Post	Pre	Post
Raspberry Pi 1	0.62	-	112	-
Raspberry Pi 2	0.03	0.05	69	155
Raspberry Pi 3	0.01	-	146	-

After recovery, the system seems to improve. Throughput rises by 70 %, losses shrink by 10%, and the end-to-end latency drops from 5.3 to 0.8 seconds. This accentuated drop could be credited to the processing capacity of the device Raspberry Pi 2, which benefits from hardware acceleration. CPU usage remains below 1 % in all periods.

This test shows that the system can recover from drops in operation, returning to a healthy state, without needing manual intervention. Nonetheless, during the recovery period, at most 430 frames were lost (43 seconds $\times$ 10 FPS), which is a period long enough to miss critical footage.

4.2.3 Stress and Load Test

The stress test looks to put high pressure on the system, to understand its limitations. In this scenario, a single processing device (Raspberry Pi #3) collects and processes frames from both cameras while they run at their maximum FPS, 50 and 60, respectively, for Raspberry Pi Camera Module 3 and Raspberry Pi AI Camera. The trace covers 2 minutes of continuous operation without topology changes.

For this test, the focus is on the following metrics: (1) **Frame processing rate** evaluates frame processing rate when system has an overflow of incoming data; (2) **End-to-end latency** tests latency variation under stressful scenarios; (3) **Data-loss ratio**, a high loss under stress indicates frames are being dropped because of the saturation; (4) **CPU/Memory usage** highlight whether an inflow of information overloads processing.

Table 5: Average captured metrics for baseline and stress tests.

	Baseline	Stress	Δ
Duration $\pm\sigma$ (s)	252.0 ± 1.11	120.9 ± 1.19	–
Frames captured	2 728	1 359	–
Frames streamed	156	70	–
Frame processing rate	0.62	0.58	–6.2%
Data-loss ratio (%)	94.3	94.8	+0.5%
End-to-end latency $\pm\sigma$ (s)	3.42 ± 0.30	6.93 ± 0.42	+103%

Table 6: Average CPU and memory usage on Raspberry Pi 3 during stress test.

	Baseline	Stress
CPU (%)	0.020	0.035
Memory (MB)	147	183

As per the statistics shown on Table 5, compared to the baseline test, the end-to-end latency is much higher, and the frame processing rate also lowers. The data loss ratio is a bit higher, at 95 %. Object detection remains the dominant compute stage, but its av-

erage time rises slightly under stress. The pure compute budget under stress conditions increases slightly. CPU usage on Raspberry Pi 3 rises marginally from 0.02 % to 0.04 %, confirming that the workload is heavily I/O-bound.

The stress test demonstrates that simply increasing camera frame-rates cannot improve analytic freshness. Higher ingress only inflates latency and maintains the same loss ratio.

4.2.4 Dynamic Topology Test

The topology test runs over twelve minutes and produces a sequence of seven fault-recovery cycles. Each cycle consists of a topology change (addition or removal of cameras/devices) followed by a new deployment. The following timeline of events was reproduced: (1) Begin with 1 Raspberry Pi (RPi 1; 0 cameras); (2) Add 1 camera (RPi 1; 1 camera); (3) Add 1 Raspberry Pi (RPi 1 + RPi 3; 1 camera); (4) Remove 1 Raspberry Pi (RPi 3; 1 camera); (5) Add 1 camera (RPi 3; 2 cameras); (6) Add 1 Raspberry Pi (RPi 2 + RPi 3; 2 cameras); (7) Remove 1 camera (RPi 2 + RPi 3; 1 camera); (8) Add 1 Raspberry Pi and 1 camera (RPi 1 + RPi 2 + RPi 3; 2 cameras).

As the focus for this test is the tolerance and adaptability of the system, only two metrics are analysed: (1) **MTTR** measures how quickly redeployments occur and how topology changes affect these values; (2) **CPU/Memory usage** ensures recovery actions do not overload nodes.

Table 7: MTTR for each cycle during topology test.

Cycle	MTTR (s)
1	41.49
2	44.56
3	41.22
4	41.72
5	73.29
6	71.84
7	43.88
Mean	51.1

Table 8: Mean CPU and memory usage during topology test.

Device	CPU (%)	Memory (MB)
Raspberry Pi 1	1.52	126
Raspberry Pi 2	0.04	218
Raspberry Pi 3	0.02	167

As Table 7 shows, recovery time remains almost constant across the seven fault-recovery cycles, with an average of 51 seconds. The orchestrator runs on an independent, separate machine, and these measurements confirm that deployment overhead does not balloon as the system grows. Most of the compute time is taken up by the task optimization (TREAO) algorithm.

Despite enduring seven rapid-fire topology changes, the system never fails to return to a steady operating state. No cascade of errors, resource exhaustion, or dead-letter queues appears in the logs, and CPU utilisation never spikes above 2%, indicating that tasks remain well within the system's capabilities.

4.3 System Analysis

Test results show the architecture keeps images anonymised at the edge, successfully recovers from node or camera loss, and stays within CPU, memory, and network capacity constraints.

Under baseline conditions, the system demonstrated stable behaviour, while resource usage remained within the constraints of the deployed hardware. The failure, stress and topology tests further confirmed the system's reliability and consistency. The ability to sustain acceptable performance levels even in partial failure scenarios strengthens the claim that decentralization can increase resilience in real-world industrial systems.

The solution, however, lacks when using hardware that has a lower processing capacity, especially during heavy computational tasks, *i.e.* human detection. The architecture, although successful, should utilize more robust devices with better capabilities, if there is a high concern for fast anonymization and low data loss rate.

5 CONCLUSION AND FUTURE WORK

This paper described a privacy-first edge/fog architecture, built on IEC 61499 Function Blocks, that can support high-resolution video streaming while being GDPR-compliant. Experimental results under nominal, failure, burst-load and dynamic-topology conditions reveal that the processing pipelines exhibit a latency below real-time thresholds requirements and can re-deploy in under a minute after device loss, while staying within the CPU and memory budgets of single-board computers. This proves that moving computation closer to the factory floor, through the utilization of edge and fog computing, helps protect personal data and reduces the amount of traffic sent over the network.

Nevertheless, the experimental results also reveal some limitations as heavy computer-vision workloads can overwhelm the Raspberry Pis if no AI accelerator is available, raising data-loss ratios. Furthermore, deployments that demand higher frame rates or faster recovery should thus pair the architecture with more capable edge hardware.

In addition, the approach described here could still benefit from future research. For example, integrating native network-discovery and priority-aware redeployment would let the orchestrator distinguish urgent failures (which risk privacy or data completeness) from benign additions that can be scheduled during quieter periods.

In summary, we have shown that it is possible to develop an edge/fog architecture that is fault-tolerant and prioritizes GDPR data-minimisation. It is supported by tools already familiar to industrial control engineers, while still allowing for future performance/feature enhancement, thus providing a solid foundation for privacy-aware, real-time analytics in future Industry 4.0 deployments.

ACKNOWLEDGEMENTS

This work was partially funded by the project "Sensitive Industry", nr. 182852, co-financed by Operational Programme for Competitiveness and Internationalization (COMPETE 2020), through national funds.

REFERENCES

Bonomi, F. and Milito, R. (2012). Fog Computing and its Role in the Internet of Things. *Proceedings of the MCC workshop on Mobile Cloud Computing*.

Dinh-Tuan, H., Beierle, F., and Garzon, S. R. (2019). MAIA: A Microservices-based Architecture for Industrial Data Analytics. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pages 23–30.

Eclipse Foundation (2024). Eclipse 4diac IDE. https://eclipse.dev/4diac/4diac_ide/. Accessed: 2025-06-02.

Etemadi, M., Ghobaei-Arani, M., and Shahidinejad, A. (2020). Resource provisioning for IoT services in the fog computing environment: An autonomic approach. *Computer Communications*, 161:109–131.

European Parliament and Council of the European Union (2025). Regulation (EU) 2016/679 of the European Parliament and of the Council.

Hu, M., Guo, Z., Wen, H., Wang, Z., Xu, B., Xu, J., and Peng, K. (2024). Collaborative Deployment and Routing of Industrial Microservices in Smart Factories. *IEEE Transactions on Industrial Informatics*, 20(11):12758–12770. Conference Name: IEEE Transactions on Industrial Informatics.

Jeschke, S., Brecher, C., Song, H., and Rawat, D. B. (2017). *Industrial Internet of Things*. Springer, First edition.

Liu, W., Huang, G., Zheng, A., and Liu, J. (2020). Research on the optimization of IIoT data processing latency. *Computer Communications*, 151:290–298.

Mahmud, R., Kotagiri, R., and Buyya, R. (2018). Fog Computing: A Taxonomy, Survey and Future Directions. In Di Martino, B., Li, K.-C., Yang, L. T., and Esposito, A., editors, *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, pages 103–130. Springer, Singapore.

Murugesan, S. (2016). Fog computing: Helping the internet of things realize its potential. *Computer*, pages 112–116.

Pereira, E. and Gonçalves, G. (2025). Online task assignment optimization in reconfigurable iec 61499-based cyber-physical production systems. TechRxiv. Preprint.

Pereira, E., Reis, J., and Gonçalves, G. (2020). Dinasore: A dynamic intelligent reconfiguration tool for cyber-physical production systems. In *Eclipse Conference on Security, Artificial Intelligence, and Modeling for the Next Generation Internet of Things (Eclipse SAM IoT)*, pages 63–71.

Pereira, E., Reis, J., Rossetti, R. J. F., and Gonçalves, G. (2024). A zero-shot learning approach for task allocation optimization in cyber-physical systems. *IEEE Transactions on Industrial Cyber-Physical Systems*, 2:90–97.

Torvekar, N. and Game, P. S. (2019). Microservices and its applications: An overview. *International Journal of Computer Sciences and Engineering*, 7(4):803–809. Accessed: 2024-12-13.

Vural, H., Koyuncu, M., and Guney, S. (2017). A systematic literature review on microservices. In *International Conference on Computational Science and Its Applications (ICCSA 2017)*, volume 10409 of *Lecture Notes in Computer Science*, pages 203–217. Springer, Cham. Accessed: 2024-12-13.

Vyatkin, V. (2016). *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*. International Society of Automation, Research Triangle Park, NC, USA, 3 edition.

Xu, L. D., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 10, NO. 4*, pages 2233–2243.