Data distribution performance for Digital Twin Synchronization: an experimental evaluation

Eduardo Freitas[1], **Assis T. de Oliveira Filho**[1],
Pedro R. X. do Carmo[1], **Marrone Dantas**[1]
Judith Kelner[1], **Djamel Sadok**[1]

[1]Grupo de Pesquisa em Redes e Telecomunicações (GPRT)
Centro de Informática (CIn) – UFPE
Recife, PE – Brazil

{eduardo.freitas, assis.tiago, pedro.carmo,}@gprt.ufpe.br

{marrone.dantas, jk, jamel}@gprt.ufpe.br

Abstract. *Digital Twin Synchronization (DTS) demands robust data distribution to handle heterogeneous flows, yet existing evaluations often overlook large messages and degraded networks. This work experimentally evaluates five middleware protocols, MQTT, AMQP, Kafka, DDS, and Zenoh, within a data-centric DTS architecture. We assess performance metrics such as latency and stability under varying message sizes (up to 2 MiB) and controlled network degradation. Our results highlight trade-offs between broker-based and brokerless architectures, identifying the most suitable solutions for the rigorous synchronization demands of Industry 4.0.*

1. Introduction

The concept of Digital Twin (DT) has emerged as a cornerstone of the Industry 4.0 paradigm, revolutionizing how physical assets are monitored, analyzed, and optimized. Although the theoretical foundations of virtual replicas are not new, recent advancements in computing power and connectivity have accelerated their adoption across domains, including smart manufacturing, predictive maintenance, and automated logistics [Fuller et al. 2020, Xu et al. 2023]. By establishing a high-fidelity virtual counterpart, industries can simulate scenarios, predict failures, and optimize operational efficiency with unprecedented precision.

While definitions of Digital Twin vary in the literature due to its diverse applicability [Liu et al. 2020], a consensus exists regarding its fundamental nature: a DT is not merely a static model but a realistic virtual replication that is continuously synchronized with a physical entity. This bidirectional relationship is the defining characteristic of the technology; the physical entity feeds status data to the virtual model, while the virtual entity processes this information to generate insights, control commands, or state updates back to the physical world.

This continuous bidirectional exchange constitutes *Digital Twin Synchronization (DTS)*. DTS acts as the system's lifeblood, ensuring that the virtual and real worlds remain coherent mirrors of one another [Rovere et al. 2018]. The synchronization flow carries payloads ranging from high-frequency telemetry to complex state vectors. Ideally, this allows for immediate actuation, such as position updates for robotics

[Liang et al. 2020], configuration changes [Zhou et al. 2023], or real-time risk warnings [do Carmo et al. 2024].

Consequently, DTS is mandatory for any robust DT application. However, implementing effective synchronization in a multifaceted industrial environment requires a compound architecture capable of handling data collection, transformation, and storage. Within this architecture, the *Data Distribution* layer enables the interoperability of all other components. It serves as the backbone responsible for transmitting data from monitoring agents to transformation engines and from decision modules back to physical actuators.

Given the heterogeneity of industrial networks and the varying requirements of data payloads from bytes to megabytes, selecting the appropriate transport mechanism is a non-trivial challenge. There is a pressing need to understand not only which communication protocols are theoretically compatible with Industry 4.0, but specifically how they perform under the rigorous constraints of DTS.

In this work, we address this gap by evaluating the leading communication protocols available in the literature, including Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), or Kafka, to perform data distribution within a DTS architecture. Unlike previous studies focused on lightweight telemetry, we assess these protocols under unique industrial conditions, including network degradation and variable payload sizes, to determine the most appropriate solution for robust Digital Twin Synchronization. Our results indicate that brokerless solutions tend to transmit at a higher rate with less delay but are highly susceptible to poor network conditions. On the other hand, the literature standards MQTT and AMQP tend to be more resilient to poor network conditions.

This article is organized as follows: Section 2 details the data-centric architecture and the rationale behind the selected middleware. Section 3 positions this study within the current state of the art. Section 4 describes the experimental methodology. Section 5 presents a comprehensive analysis of the obtained results. Finally, Section 6 offers concluding remarks and outlines directions for future research.

2. Background

Digital Twin Synchronization (DTS) is the cornerstone of a proper DT deployment. Without it, a DT remains a straightforward simulation of an industrial plant, behaving strictly according to pre-programmed logic. In such a static scenario, updates from the real world would require manual input, rendering simulation results useful merely as insights rather than enabling automatic actuation on real-life entities. Consequently, addressing how to perform DTS properly is a fundamental requirement for Industry 4.0.

2.1. The Data-Centric Architecture

To address this challenge, our previous work [Freitas et al. 2026] proposes a unified *Data-Centric Industrial Digital Twin Synchronization Architecture*. This reference architecture, illustrated in Figure 1, presents synchronization components with specific roles that form a complete framework.

The **Data Collection** is the first component to interact with the industry fabric. As the name implies, it collects data from the entire ecosystem, including physical devices

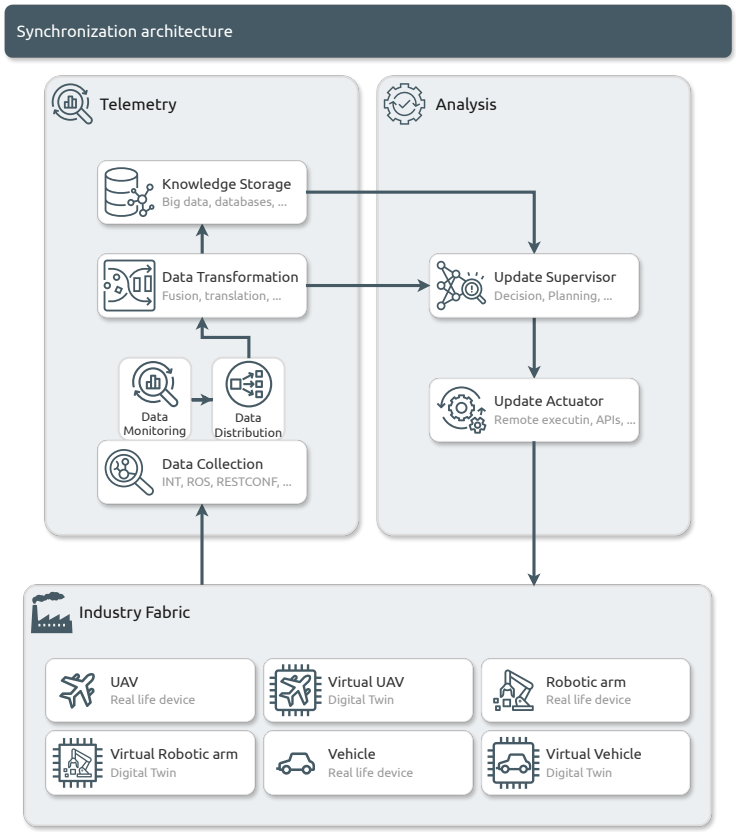

Figure 1. Digital Twin Synchronization Architecture.

(e.g., robotic arms, cameras) and their virtual counterparts. It constitutes two major sub-components:

1. **Data Monitoring:** Responsible for fetching data from all devices using manifold protocols suitable for each device's nature, such as In-band Network Telemetry (INT) or SensorThings API.
2. **Data Distribution:** This sub-component is significant. It transmits data gathered by the monitoring component to other appropriate components.

Although represented as a sub-component of collection, **Data Distribution** is omnipresent in the architecture. Whether transmitting data from monitoring to transformation, transformation to knowledge storage, or transformation to the update supervisor, all data exchange relies on this layer.

The traffic overhead of standard poll-based data transmission, which is typically associated with more conventional network management protocols like SNMP, can harm data distribution performance and cause more interference than is necessary [Naik 2017]. The latency caused by modifications on the network can be decreased by publishing the gathered data in a stream and subscribing to it [Song et al. 2022]. The Message-oriented middleware is useful because it can connect different devices and send information from monitoring agents to data transformation parts in a way that maximizes bandwidth efficiency and minimizes energy use [Gemirter et al. 2021]. Numerous publish/subscribe middleware solutions are available, including Zenoh, Kafka, Data Distribution Service (DDS), AMQP, and others. To determine an appropriate implementation through a thor-

ough evaluation, it is crucial to consider parameters such as the number of devices on the scene, the type of messages sent, and the available resources [Gamess et al. 2021] [Akasiadis et al. 2019].

Because of this, we provide an evaluation of the most important middleware protocols available in the literature, namely MQTT, AMQP, DDS, Kafka, and Zenoh. We provide in the following section an overview of related works and why the chosen middleware protocols are prevalent in the literature.

3. Related Works

DTS imposes communication requirements that transcend traditional IoT telemetry, demanding continuous and bidirectional state mirroring between physical and virtual domains. While recent literature has been prolific in defining architectural models and synchronization patterns, experimental validation of the transport layer under adverse conditions combining network degradation with divergent message sizes remains a gap. This section analyzes the state of the art from three perspectives: synchronization architectures, industrial middleware evaluation, and robustness in edge computing environments.

3.1. Synchronization architectures

The need for precise synchronization is the fundamental consensus in Digital Twin research. [Akbarian et al. 2020] demonstrates that synchronization quality directly impacts control errors in industrial systems, proposing distinct architectures to mitigate state divergences. Expanding this view, [Ferko et al. 2022] perform a systematic mapping of 140 studies, identifying a predominance of layered and service-oriented patterns. However, they note that performance efficiency is often treated as an abstract quality attribute rather than a subject of deep transport layer validation.

Recent proposals focus on the semantic and functional structuring of DTs. [Somma et al. 2025] introduces *TwinArch*, a domain-independent reference architecture, while [Yu et al. 2025] proposes a multidimensional model for digital workshops, emphasizing dynamic virtual-physical interaction. [Alghamdi 2024] advances the taxonomy by cataloging synchronization patterns (e.g., time-driven, event-driven) for different domains. Similarly, [Dihan et al. 2024] analyzes DTs through the lens of the data lifecycle, highlighting the challenge of handling large volumes and heterogeneity.

However, practical implementations of these architectures typically assume ideal or simplified network conditions. [Caiza and Sanz 2023] implement functional industrial DTs using OPC UA and Ethernet, reporting communication times in the order of 100 ms. While validating functional integration, these works do not stress the infrastructure with large messages or network instability. Even when security and state replication are the focus, as in [Gehrmann and Gunnarsson 2020], performance evaluation remains limited, without exploring throughput or latency limits under congestion. In summary, the architectural literature defines *what* must be synchronized but rarely validates *how* the infrastructure supports this synchronization under real-world stress.

3.2. Middleware evaluations

To materialize synchronization, the industry relies on standardized middleware. Comparative studies, such as those by [Yoshino et al. 2021] and [Silva et al. 2021], have established important baselines for Industrial Internet of Things (IIoT), demonstrating that

MQTT is efficient for small messages, while AMQP offers greater reliability, and CoAP provides lower time-to-completion. The work by [Wytrebowicz et al. 2021] complements this view with a functional analysis of these protocols for IoT devices.

In the context of real-time systems, [Peeroo et al. 2023] and [Solpan and Kucuk 2022] highlight DDS (and DDS-XRCE) for its configurable QoS capabilities. However, [Gavrilov et al. 2022] warns that although RTPS (DDS) handles complex traffic better than MQTT, such performance is not guaranteed for large messages. Recently, next-generation protocols like Zenoh have demonstrated superior performance. [Liang et al. 2023] and [Zhang et al. 2023] show that Zenoh outperforms Kafka and DDS in throughput and latency, presenting lower variations in error in Wi-Fi and 4G networks.

Indeed, a methodological limitation persists across most of these evaluations: the focus on "lightweight telemetry". [Mishra et al. 2021] performs stress tests on MQTT brokers, focusing only on message rate, not size. Even [Filho et al. 2022], when evaluating image transmission, focuses on decision time rather than continuous state synchronization. Consequently, the behavior of these middlewares when transmitting massive state vectors (e.g., 2 MiB), typical of a visual or analytical DT, remains underanalyzed.

3.3. Network degradation and edge robustness

The final factor, often ignored in laboratory settings, is the variability of the Edge environment. [Dizdarevic et al. 2023] provides experimental evidence that broker-based architectures degrade rapidly under packet loss and latency, creating centralized bottlenecks. Additionally, [Bezerra et al. 2024] shows that security mechanisms amplify this impact, making latency fundamental.

In contrast, brokerless architectures appear more resilient. [Chovet et al. 2024] indicates that Zenoh maintains connectivity and low latency in unstable mesh networks where traditional DDS fails. [Corte et al. 2025] reinforce this through systematic evaluations, showing that brokerless libraries eliminate single points of failure and offer higher raw throughput.

3.4. Synthesis and contribution

The literature presents a division: on one side, intelligent DT architectures that abstract communication; on the other, middleware evaluations that focus on light loads or ideal network conditions. There is a lack of integrated validation considering the hostile reality of the factory floor.

This work fills this gap by conducting an experimental evaluation that:

1. **Combines architectural and network views:** Evaluates whether middlewares (MQTT, AMQP, Kafka, DDS, Zenoh) support the synchronization requirements defined by reference architectures [Akbarian et al. 2020, Ferko et al. 2022, Freitas et al. 2026].
2. **Scales the message size:** Tests messages up to 2 MiB, simulating the synchronization of complex states neglected by telemetry evaluations.
3. **Emulates reality:** Introduces controlled degradation (25 ms delay, 5% loss), testing distribution robustness where it is most relevant.

4. Experiment Setup

To evaluate the performance of data distribution protocols, we built a testbed consisting of three KVM Virtual Machines (VMs). These three VMs are connected via a virtual switch that isolates them from any other network communication possible inside the server. The isolated virtual network has a 1 GbE speed. The host machine where the VMs run has a 12 core Intel Core i7-5820K CPU with 3.30 GHz, 12 GiB of RAM, running Proxmox Virtual Environment 8.2.4, with kernel 6.2.16. All the VMs have 1 vCPU and 1 GiB of RAM.

The experiment consists of each VM developing a role in the publish/subscribe communication pattern. The first VM is the publisher, the second is the subscriber, and the last one is the broker. In protocols that do not need a broker, such as DDS or Zenoh, the third VM is not used. In each VM, there is a variable number of agents that fill the role designated by the VM (i.e., the agents in the subscriber VM are all subscribers). Figure 2 illustrates the experiment topology, with how each VM connects itself.

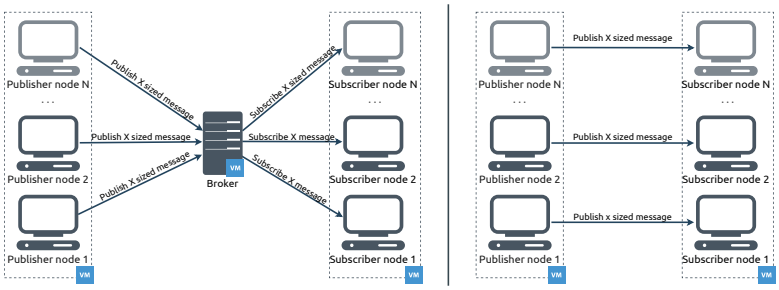

Figure 2. Experiments testbed

Since we want to create a scenario that resembles a possible DTS of an industrial plant, we create different message sizes to send during the experiment. Each VM has a pair of agents, where each agent publishes and subscribes to a specific message size, either 32 KiB or 2 MiB. We chose the message sizes to better represent the variety of possible synchronization data in an industrial DTS environment, such as small IoT devices synchronizing sensor data or robots/cameras synchronizing large images. The message size is referenced in the literature, in works like [Liang et al. 2023, Gamess et al. 2021]. To further expand the industrial scenario, the number of agent pairs also changes, with a total of either 12, 36, or 108 agents. The agent number is also paired to better emulate a complex industrial scenario where multiple industrial entities are acting upon the DT, such as cameras, sensors, robots, and network equipment. All publishers send their defined message size with a 1-second interval between each transmission and send a total of 10 messages.

The agents used to publish and subscribe messages are implemented in Python, using the appropriate libraries. For AMQP we used `pika` library with RabbitMQ broker. For DDS we used `cyclonedds` library. For Kafka we used `kafka` library and broker For MQTT we used `paho` library with RabbitMQ broker. And for Zenoh we used `zenoh` itself for library. We used the same script, changing only the necessary calls for the library's publish and subscribe method.

To resemble the network constraints possible in an industrial scenario, we limit the network conditions in our experiments, creating three scenarios:

- No network limitation, where there is no limitation in the 1 GbE network link between all VMs,
- Moderate network limitation, where the link between the VMs has a 5 ms delay, bandwidth reduced to 200 Mbps, and a loss rate of 0.1%,
- High network limitation, where the link has a 25 ms delay, 100 Mbps bandwidth, and a 5% loss rate.

The metrics that we collect and analyze are publication Round-Trip Time (RTT), subscription delay and publisher CPU utilization. The publication RTT refers to the latency time starting when the publisher publishes a message in the queue, the subscriber receives this message and publishes back a confirmation in a different queue, which the publisher is subscribed to, and receives the confirmation. This metric is referenced in the literature and can be found in works like [Luzuriaga et al. 2015, Gamess et al. 2021]. Since the time to transmit these messages can vary according to middleware and network limitations, we collect the delay by the message sequence instead of merely the experimentation time. In our scenario, experiment duration ranged from 50 to 140 seconds.

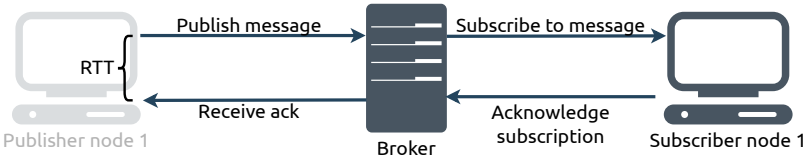

Figure 3. Representation of how we calculate publication RTT.

The subscription delay is the delay for a subscriber receiving one message and the next one. The time in our experiment should always be 1 second, considering a perfect scenario, since we send a message every 1 second. The subscription delay, often called subscription jitter, is also a referenced metric, present in works like [Gamess et al. 2021, Liang et al. 2023].

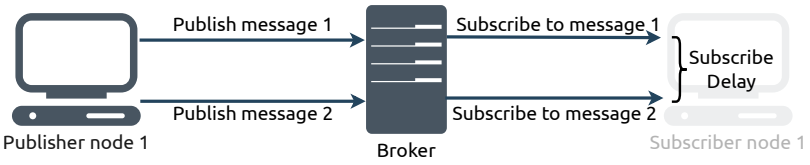

Figure 4. Representation of how we calculate subscription delay.

5. Results and Discussion

In this section, we present the experimental results obtained from the testbed described in Section 4. The analysis is structured to evaluate the trade-offs between middleware type, network limitations, and packet size. All results show the mean and confidence interval of 95%. All experiment scenarios were repeated 30 times.

Figures 5 and 6 show the results for the RTT for the agents that send 32 KiB and 2 MiB messages, respectively, on any network limitation and agent number. Zenoh and DDS are able to keep low RTT, showing the prevalence of not needing a broker to forward messages. With low number of agents and no network limitation, Zenoh can keep

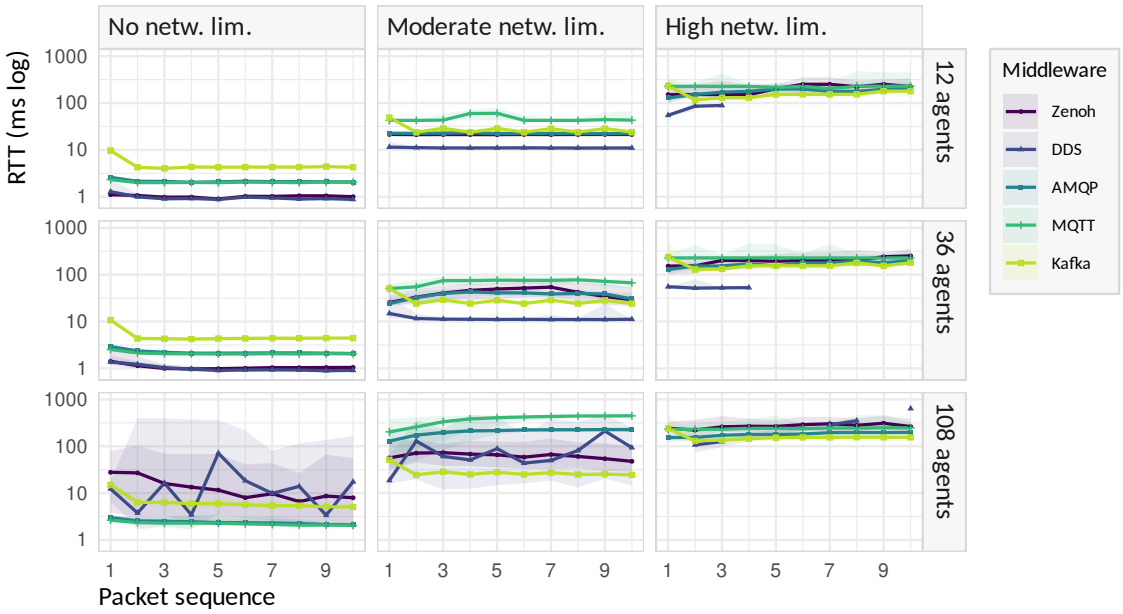

Figure 5. RTT results for agents sending 32 KiB messages, separated by network limitation and number of agents.

the lowest RTT, consistently below 10 ms. In addition, the difference between 12 and 36 agents is small.

In low-scale scenarios (12 and 36 nodes), brokerless protocols (Zenoh and DDS) exhibit the lowest latency for small messages. This confirms the architectural advantage of eliminating the central broker hop in ideal conditions. However, a divergence occurs with larger messages (2 MiB): only Zenoh maintains low latency and stability. DDS shows increased jitter, likely due to the overhead of fragmenting large messages over DDS's underlying transport protocol, without a TCP flow control.

Kafka consistently presents the highest latency values across all packet sizes. Furthermore, it fails to deliver the 2 MiB messages successfully in this configuration. This behavior aligns with Kafka's design as a high-throughput, log-based storage system rather than a real-time, low-latency transport. Its batching mechanisms and disk-persistence overhead, while beneficial for "Knowledge Storage", prove detrimental for the immediate synchronization loop.

When more agents are present in the network, the results change. As the node count scales to 108, DDS exhibits severe instability. This is characteristic of the discovery traffic explosion inherent in full-mesh peer-to-peer architectures ($O(N^2)$ complexity). Conversely, the broker-based protocols (AMQP and MQTT) maintain constant, albeit higher, latency values, demonstrating that the broker acts as a stabilizer for control plane traffic as the system scales. However, while Zenoh is less unstable than DDS, its latency surpasses MQTT/AMQP in this high-density scenario, suggesting that its hybrid routing algorithms face overheads under dense clustering.

With the introduction of packet loss (0.1%), the performance hierarchy shifts. MQTT exhibits unexpectedly high latency even for small packets, indicating a sensitivity to TCP retransmission timeouts in the presence of jitter. Both Kafka and DDS fail to

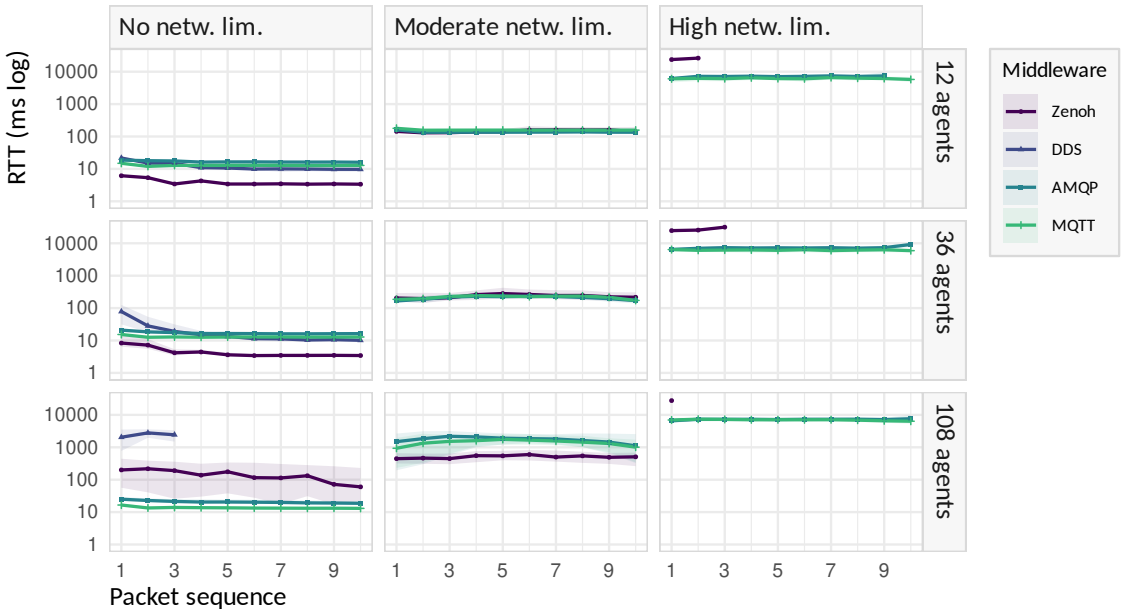

Figure 6. RTT results for agents sending 2 MiB messages, separated by network limitation and number of agents.

transmit 2 MiB packets entirely.

The most significant finding in this scenario is that *only MQTT and AMQP successfully delivered 2 MiB packets.* While the latency is high ($\approx$ 10 seconds), the delivery is guaranteed. This resilience is attributed to the robust TCP flow control and buffering mechanisms inherent in the broker architecture, which decouples the publisher from the subscriber, absorbing network variations. In contrast, DDS, Zenoh, and Kafka failed to synchronize large states under 5% packet loss, rendering them unsuitable for large-state synchronization (e.g., visual data) in hostile edge environments without extensive tuning.

The subscriber delay, shown in Figures 7 and 8, shows that jitter is not as high as the RTT, except for the DDS case. All middlewares had constant delivery intervals around 1 s, except DDS with high number of agents. In this scenario, DDS could not deliver more than 3 messages and also had high RTT. In addition, a specific case for 108 agents and moderate network limitation is that MQTT had a slowly increasing delay, which got close to 2 s.

With the larger messages, the results converge with the RTT ones, showing that AMQP and MQTT are the only ones to maintain a steady rate with high network limitations. The average delivery delay is 10 s for these middlewares, which are the only ones to successfully deliver the messages.

The CPU results shown in Figure 9 corroborate the latency findings regarding scalability.

- **DDS Saturation:** DDS quickly reaches 100% CPU usage. At 108 nodes, this usage does not reduce, indicating a continuous struggle to maintain discovery state and mesh synchronization. This confirms that untuned DDS is computationally unsuitable for high-density DTS. Furthermore, using DDS could result in poor scalability considering industrial DTS, especially when limited hardware can also

Figure 7. Subscriber delay results for agents sending 32 KiB messages, separated by network limitation and number of agents.

be used.

- **Broker Efficiency:** AMQP and MQTT show a significant usage peak during transmission followed by an immediate drop to idle. By offloading routing logic to the central broker, the edge agents remain lightweight, a feature for battery-powered industrial IoT devices.
- **Zenoh's Profile:** Zenoh exhibits a "spread out" usage pattern, taking longer to return to idle. While lighter than DDS, its hybrid routing requires more sustained processing than the simple client-server model of MQTT.

5.1. Synthesis

The results validate the hypothesis that no single protocol satisfies all DTS requirements. While Zenoh and DDS offer superior latency for real-time control in networks with higher capacity, they struggle to manage heavy messages and network degradation. Conversely, AMQP and MQTT act as the "tanks" of the architecture, sacrificing rate and latency for the guarantee of delivery, proving to be the only viable options for synchronizing larger Digital Twin states in unstable industrial environments.

6. Conclusion

Digital Twin Synchronization is often idealized in the literature as a seamless flow of data. However, our experimental investigation reveals that the harsh reality of industrial networks, characterized by packet loss, jitter, and heterogeneous message sizes, imposes severe constraints that no single middleware protocol can satisfy in isolation.

By subjecting five distinct protocols (MQTT, AMQP, Kafka, DDS, and Zenoh) to a unified Data-Centric Architecture testbed, we identified an important split between *performance* and resilience. In ideal or local network conditions, brokerless architectures (specifically Zenoh and DDS) demonstrated superior efficiency, offering the lowest

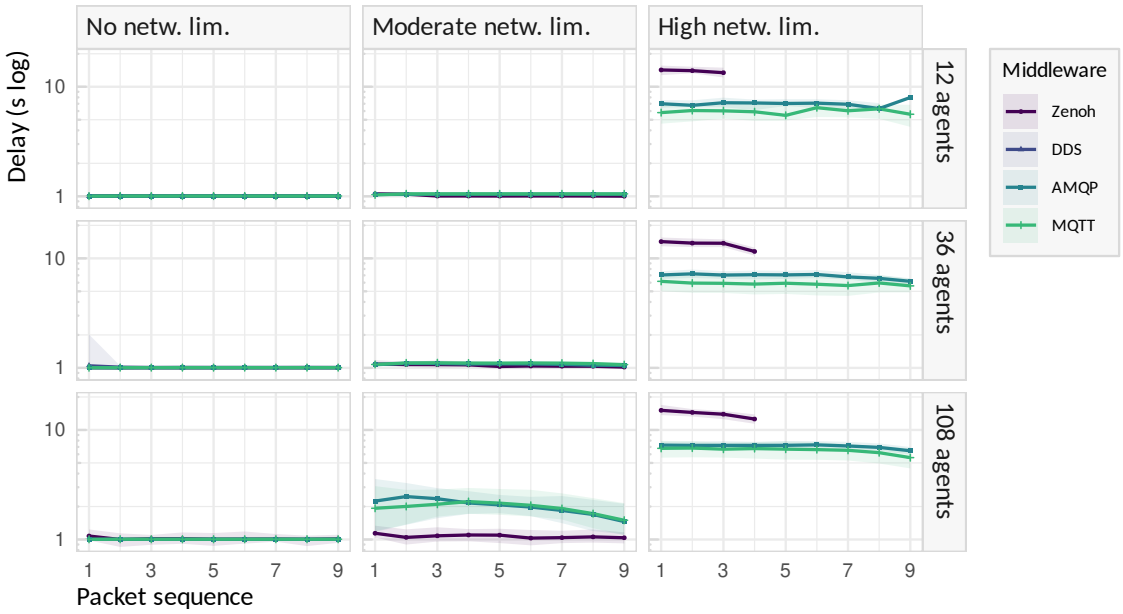

Figure 8. Subscriber delay results for agents sending 2 MiB messages, separated by network limitation and number of agents.

latency for telemetry synchronization. This confirms their suitability for real-time, intra-cell robotic control where network stability is guaranteed.

However, the introduction of realistic edge constraints coupled with large messages alters the landscape. Our results indicate that high-performance peer-to-peer discovery mechanisms (inherent to DDS) collapse under network stress and high node density (108 nodes), leading to CPU saturation and transmission failure. In contrast, the reliable, TCP-backed broker architectures of MQTT and AMQP proved to be the only viable solutions for synchronizing complex Digital Twin states in degraded environments. Although they incur higher latency, their ability to guarantee delivery where others fail makes them indispensable for critical system observability.

Therefore, we conclude that a robust DTS implementation necessitates a *hybrid leveled architecture:*

1. **Local/Real-Time:** Utilization of Zenoh or DDS for high-frequency, low-latency control loops within the local network segment.
2. **Global/Resilient:** Deployment of AMQP or MQTT as the backbone for transporting heavy state updates and critical commands across unreliable or unpredictable network conditions.

Future work will focus on implementing and validating this hybrid protocol stack within the reference architecture, specifically analyzing the computational overhead of protocol bridging and the impact of transport-layer security (TLS) on synchronization latency.

Acknowledgment

This research was supported by *Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)*, *Fundação de Amparo a Ciência e Tecnologia de Pernambuco (FACEPE)*,

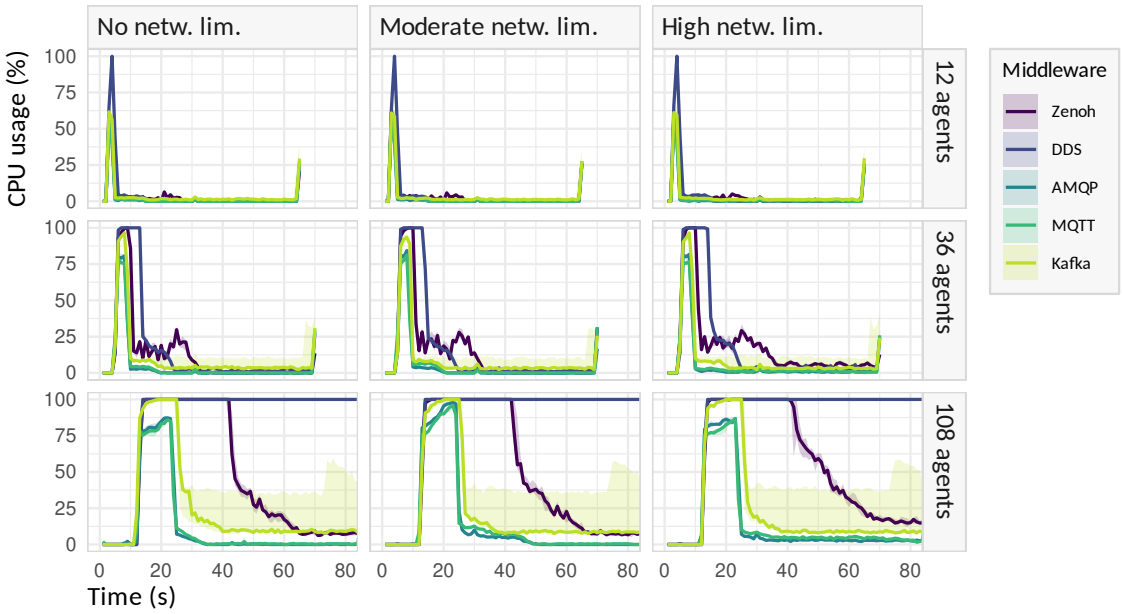

Figure 9. CPU usage for each middleware, separated for each number of agents and network conditions.

Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and Ericsson Telecomunicações S.A. Brazil.

References

Akasiadis, C., Pitsilis, V., and Spyropoulos, C. D. (2019). A Multi-Protocol IoT Platform Based on Open-Source Frameworks. *Sensors*, 19(19).

Akbarian, F., Fitzgerald, E., and Kihl, M. (2020). Synchronization in Digital Twins for Industrial Control Systems. *ArXiv*, abs/2006.03447.

Alghamdi, W. (2024). Synchronization Patterns for Digital Twin Systems. *Journal of Applied Data Sciences*.

Bezerra, W. R., Koch, F., and Westphall, C. B. (2024). Performance of security options for message protocols: A comparative analysis. *International Journal of Network Management*, 34.

Caiza, G. and Sanz, R. (2023). Digital Twin to Control and Monitor an Industrial Cyber-Physical Environment Supported by Augmented Reality. *Applied Sciences*.

Chovet, L., Garcia, G. M., Bera, A., Richard, A., Yoshida, K., and Olivares-Méndez, M. (2024). Performance Comparison of ROS2 Middlewares for Multi-robot Mesh Networks in Planetary Exploration. *Journal of Intelligent & Robotic Systems*, 111.

Corte, L. L., Rashid, S. A., and Dan, A.-M. (2025). Performance Evaluation of Brokerless Messaging Libraries. *ArXiv*, abs/2508.07934.

Dihan, M. S., Akash, A. I., Tasneem, Z., Das, P., Das, S. K., Islam, M. R., Islam, M. M., Badal, F., Ali, M. F., Ahmed, M. H., Abhi, S. H., Sarker, S., and Hasan, M. (2024). Digital twin: Data exploration, architecture, implementation and future. *Heliyon*, 10.

Dizdarevic, J., Michalke, M., and Jukan, A. (2023). Engineering and Experimentally Benchmarking Open Source MQTT Broker Implementations. *ArXiv*, abs/2305.13893.

do Carmo, P. R., de Freitas Bezerra, D., Filho, A. T. O., Freitas, E., Silva, M. L., Dantas, M., Oliveira, B., Kelner, J., Sadok, D. F., and Souza, R. (2024). Living on the edge: A survey of Digital Twin-Assisted Task Offloading in safety-critical environments. *Journal of Network and Computer Applications*, 232:104024.

Ferko, E., Bucaioni, A., and Behnam, M. (2022). Architecting Digital Twins. *IEEE Access*, PP:1–1.

Filho, A. O. O., Barbosa, G. B. N., Rodrigues, I., Cani, C., Kelner, J., Sadok, D., and Souza, R. S. (2022). Experimental network performance evaluation for human-robot interaction collision detection using cameras. *Research, Society and Development*.

Freitas, E., de Oliveira Filho, A. T., do Carmo, P. R. X., Sadok, D., and Kelner, J. (2026). Digital Twin Synchronization: towards a data-centric architecture. *arXiv*, abs/2601.23051.

Fuller, A., Fan, Z., Day, C., and Barlow, C. (2020). Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, 8:108952–108971.

Gamess, E., Ford, T. N., and Trifas, M. (2021). Performance Evaluation of a Widely Used Implementation of the MQTT Protocol with Large Payloads in Normal Operation and under a DoS Attack. In *Proceedings of the 2021 ACM Southeast Conference*, ACM SE '21, page 154–162, New York, NY, USA. Association for Computing Machinery.

Gavrilov, A., Bergaliyev, M., Tinyakov, S., Krinkin, K., and Popov, P. (2022). Using IoT Protocols in Real-Time Systems: Protocol Analysis and Evaluation of Data Transmission Characteristics. *J. Comput. Networks Commun.*, 2022:7368691:1–7368691:18.

Gehrmann, C. and Gunnarsson, M. (2020). A Digital Twin Based Industrial Automation and Control System Security Architecture. *IEEE Transactions on Industrial Informatics*, 16:669–680.

Gemirter, C. B., Senturca, C., and Baydere, S. (2021). A Comparative Evaluation of AMQP, MQTT and HTTP Protocols Using Real-Time Public Smart City Data. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 542–547.

Liang, C.-J., McGee, W., Menassa, C., and Kamat, V. (2020). Bi-Directional Communication Bridge for State Synchronization between Digital Twin Simulations and Physical Construction Robots. In *Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 1480–1487, Kitakyushu, Japan. International Association for Automation and Robotics in Construction (IAARC).

Liang, W.-Y., Yuan, Y., and Lin, H.-J. (2023). A Performance Study on the Throughput and Latency of Zenoh, MQTT, Kafka, and DDS. *ArXiv*, abs/2303.09419.

Liu, M., Fang, S., Dong, H., and Xu, C. (2020). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*.

Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., and Manzoni, P. (2015). A comparative evaluation of AMQP and MQTT protocols over unstable and mobile

networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 931–936.

Mishra, B., Mishra, B., and Kertész, A. (2021). Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies*.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7.

Peeroo, K., Popov, P., and Stankovic, V. (2023). A Survey on Experimental Performance Evaluation of Data Distribution Service (DDS) Implementations. *ArXiv*, abs/2310.16630.

Rovere, D., Pedrazzoli, P., dal Maso, G., Alge, M., and Ciavotta, M. (2018). *The Digital Shopfloor - Industrial Automation in the Industry 4.0 Era*, chapter A Centralized Support Infrastructure (CSI) to Manage CPS Digital Twin, towards the Synchronization between CPS Deployed on the Shopfloor and Their Digital Representation, page 19. Imprint River Publishers.

Silva, D. M. A., Carvalho, L., Soares, J. A. M., and Sofia, R. C. (2021). A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA. *Applied Sciences*, 11:4879.

Solpan, S. and Kucuk, K. (2022). DDS-XRCE Standard Performance Evaluation of Different Communication Scenarios in IoT Technologies. *EAI Endorsed Transactions on Internet of Things*.

Somma, A., Amalfitano, D., Benedictis, A. D., and Pelliccione, P. (2025). TwinArch: A Digital Twin Reference Architecture. *ArXiv*, abs/2504.07530.

Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and Wang, A. (2022). Network Telemetry Framework. RFC 9232.

Wytrebowicz, J., Cabaj, K., and Krawiec, J. (2021). Messaging Protocols for IoT Systems—A Pragmatic Comparison. *Sensors (Basel, Switzerland)*, 21.

Xu, H., Wu, J., Pan, Q., Guan, X., and Guizani, M. (2023). A Survey on Digital Twin for Industrial Internet of Things: Applications, Technologies and Tools. *IEEE Communications Surveys & Tutorials*, pages 1–1.

Yoshino, D., Watanobe, Y., and Naruse, K. (2021). A Highly Reliable Communication System for Internet of Robotic Things and Implementation in RT-Middleware With AMQP Communication Interfaces. *IEEE Access*, 9:167229–167241.

Yu, J., Chen, C., Zhang, C., and Ji, W. (2025). Real-Time Monitoring and Dynamic Interaction Methods Based on Digital Twin Workshop Theory. *Processes*.

Zhang, J., Yu, X., Ha, S., Queralta, J. P., and Westerlund, T. (2023). Comparison of DDS, MQTT, and Zenoh in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems. *ArXiv*.

Zhou, C., Chen, D., Martinez-Julia, P., and Ma, Q. (2023). Data Collection Requirements and Technologies for Digital Twin Network. Internet-Draft draft-zcz-nmrg-digitaltwin-data-collection-03, Internet Engineering Task Force. Work in Progress.