

## RESEARCH

# A Cloud-Fog Computing Architecture for Real-Time Digital Twins

Francisco P. Knebel\*, Juliano A. Wickboldt and Edison P. de Freitas

## Abstract

Digital Twin systems are designed as two interconnected mirrored spaces, one real and one virtual, each reflecting the other, sharing information, and making predictions based on analysis and simulations. The correct behavior of a real-time Digital Twin depends not only on the logical results of computation but also on the timing constraints. To cope with the large amounts of data that need to be stored and analyzed, modern large scale Digital Twin deployments often rely on cloud-based architectures. A significant portion of the overall response time of a Digital Twin is spent on moving data from the edge to the cloud. Therefore, implementing Digital Twins using cloud-fog architectures emerges as an alternative to bring computing power closer to the edge, reducing latency and allowing faster response times. This paper studies how suitable the use of a cloud-fog architecture is to handle the real-time requirements of Digital Twins. Based on a realistic implementation and deployment of Digital Twin software components, it is possible to conclude that the distribution of Digital Twins in a fog computing setup can reduce response times, meeting its real-time application requirements.

**Keywords:** Digital Twin; Real-Time Systems; Cloud Computing; Fog Computing

## 1 Introduction

The Digital Twin concept was first introduced in 2002 as a model of a system containing two interconnected mirrored spaces, one real and one virtual, each space reflecting the other while sharing real-time information [1]. Digital Twins must be designed as real-time systems so that the virtual part can reflect the real one accurately. The correctness of the system behavior depends not only on the logical results but also on the timing constraints defined for this system [2].

Essentially, a Digital Twin is a computer program that acquires real-world data about a physical system and generates simulations on how that system will be affected by this data. Long before the terminology, Digital Twins were used in action by NASA, in order to monitor and maintain machines in outer space, a situation where hands-on contact with devices would be impossible, allowing for remote diagnosis and fixing problems that could occur [3]. Digital Twins can be defined as model-based representations of real-time systems, as long as they are adaptable, non-static, systems, which evolve over time with their environment [4].

To reflect their physical counterparts, Digital Twins are composed by a large number of sensors, responsible for acquiring the used data. The rapid expansion of sensors powered by the Internet of Things (IoT) is what makes Digital Twins possible, but with this increased data generation, centralized applications oftentimes experience high latency and poor connectivity situations which affects the performance of applications. With the advent of IoT devices and its increased data volume in cloud systems, moving all data to the cloud does not satisfy time-sensitive application requirements [5]. Fog computing enables computing, storage, networking, and data management on network nodes in closer proximity to edge devices, allowing the processing to happen not only in the cloud, but also between the edge and the cloud [5], besides on the edge [6].

Applications in several areas can be improved with the usage of fog computing, including smart industries [7, 8], homes [9], cities [10, 11] and healthcare [12], all real-time systems with time-critical processing to avoid accidents and to provide a better and more reliable experience without suffering from cloud latency. Integrating fog computing with IoT and Digital Twins enables mirroring physical and virtual spaces while

\*Correspondence: [francisco.knebel@inf.ufrgs.br](mailto:francisco.knebel@inf.ufrgs.br)

Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, BR

Full list of author information is available at the end of the article

achieving the necessary timing requirements of real-time systems.

Among the many challenges of implementing a fog computing network, computation offloading and data preprocessing are critical in terms of scalability and guarantee of real-time requirements [13]. Balancing the computational workload between multiple fog nodes, closer to the edge devices, enables the network to be more efficient in handling timing critical tasks, by distributing operations between different devices and reducing communications between edge and cloud. For Digital Twins operate under time-sensitive conditions, research is still required so that a number of connected services and devices can scale and correctly execute real-time operations in timely fashion. It is still challenging to manage, process, and store all this data in a highly distributed environment [14, 7].

Literature related to the usage of cloud and fog in IoT-related themes is already existent and discussed, but are limited to abstract and conceptual proposals for these architectures, lacking on practical demonstrations, which show absolute differences between the advantages of each deployment scenario. In light of these challenges and possibilities, this paper proposes the usage of a cloud-fog architecture that can manage a large scale number of real-time Digital Twins, increasing performance while reducing latency, to meet timing requirements. Thus, the contribution of this work is the study of an distributed architecture to support Digital Twin applications that must meet real-time requirements, avoiding timing problems that usually happen in centralized approaches.

The remainder of this paper is structured as follows. Section 2 provides background information for Digital Twin and cloud, fog, and edge computing concepts. Section 3 discusses related work. Section 4 describes the proposed solution. Section 5 presents the experiments, the methodology, and the obtained results. Section 6 discusses the results obtained in the experimental scenarios. Section 7 concludes the paper and presents directions for future work.

## 2 Background

### 2.1 Digital Twins

Digital Twins are systems designed to work with rich data provided by both the insight of the models and implicit knowledge of objects, products of their behavior, only requiring the availability of raw data about their physical counterparts. To achieve this, Digital Twins must accurately simulate, analyze, and predict real-world events and situations, via the collection of real-time physical data, mirroring it into its virtual space. Digital Twins can not only anticipate and predict but can issue proactive measures to avoid

problems by combining real-time data with the knowledge of other Digital Twins instances. Simulations can forecast the outcomes, allowing human operators to choose, knowing the consequences of the possible actions, and set the optimal solution [15].

The implementation of self-aware Digital Twins creates modern control systems that adapt to their environment, not only mimicking but understanding the reasons behind the behavior of their physical counterparts [16]. The virtual part of a Digital Twin system is not necessarily physically implemented along with the native physical devices. Conversely, it can be implemented physically apart from its real counterpart in a distributed and evolving manner, dynamically learning patterns, and helping predict possible disruptions throughout the life of the system [17].

### 2.2 Cloud, Fog, and Edge Computing

Cloud computing enables convenient, on-demand access to a shared pool of computing resources [18]. Cloud computing transfers the responsibility of data processing and storage to remote data centers, via the Internet. It allows resources to be geographically distributed, increasing data redundancy and reliability [19, 20]. Fog nodes are distributed computing entities, formed by at least one networked physical processing device, able to execute distributed tasks [21, 22]. Edge computing moves processing nearest to where it is needed, allowing computation closer to the source, reducing cloud traffic and service latency, improving response times [23].

Fog differentiates itself from the cloud due to its nodes being deployed in large numbers at less centralized locations, compared to centralized cloud data centers. In essence, fog computing is an extension of cloud computing into the physical world and their multi-connected devices, ensuring data availability where and when it is needed [24], while edge computing represents the nearest the processing resources can be placed regarding the data source. Positioning of the network end-devices in near proximity of the user or application increases the response speed, via minimizing network latency [25, 26].

This work presents the idea of using both cloud and fog layers, but excluding edge. Since the edge is composed mostly of devices without complete control from the cloud provider, concerns are presented in regards to system reliability and availability. It can be used to reduce latency and enable more computing power, but due to its possible short and temporary availability, using a more controlled layer, in this case, the fog enables the objective of latency reduction while still being supervisable.

### 3 Related Work

Many studies combining cyber-physical systems, Digital Twins, and fog computing exist, but to the best of our knowledge, not in the context of real-time systems. Cyber-physical systems using Digital Twins in the cloud, fog, or edge environments have been proposed by Qi *et al.* [27] and Kim [28] as three levels of systems: unit, system, and system of systems (or service) level. The proposal by Qi *et al.* [27] highlights that cloud computing enables on-demand resources and computational sharing, while fog computing shifts computation, storage, and networking of the cloud to the edge network, and edge computing allows processing at closer proximity to the data source, which can benefit cyber-physical systems. Kim [28] states that the key driver to moving towards edge/fog computing is time-sensitive communication, which is a required feature for real-time systems. It also mentions that distinctions between the edge, fog, and cloud layer need to be further clarified to be better explored in different setups.

A decentralized model for an edge-fog network with IoT devices is proposed by Mohan and Kangasharju [29]. The proposal has the edge built with voluntary and loosely coupled general-purpose devices, one or two hops from the IoT resources, and horizontal connection between the edge devices. The fog layer, in this model, is a consolidation of networking devices (*i.e.*, routers, switches) where edge devices can offload computationally intensive tasks. The model includes a cloud component, but it is used only for centralized data storage, with no computational capabilities. This approach would not suffice in a scenario with high computational requirements, since it has no fallback to the cloud, in cases where the edge and fog are overloaded with tasks.

The benefits of fog and edge computing, which include low latency, locality, and scalability, are well known. However, Ahmad and Afzal [30] state that although there are solutions for this domain, many aspects are still unexplored in practical scenarios, lacking an architecture that fully implements these principles. Despite this landscape, there are examples of real situations where fog computing was used to build cyber-physical systems [31], to accelerate production processes. In the scenario described by Fernández-Caramés *et al.* [31], experiments revealed that fog nodes executed their tasks faster than the same tasks in a purely cloud-dependent environment. The authors report that these nodes can process more data while under high-load situations, in which real-time processing is constrained.

In short, given the studies presented above, currently, the Digital Twin literature still has many unexplored study areas, lacking on an actual deployment

of a real implementation of Digital Twins which utilizes the benefits of a cloud-fog architecture, presenting how it was built, its components and with experiment results proving its benefits.

### 4 Proposal

This section initially describes in detail the problem addressed in this work. Then, it provides an overview of how the Digital Twin architecture was designed, its components, and details on how each of the components is used in the experiments presented in Section 5.

#### 4.1 Addressed Problem

The literature review revealed that the use of cloud and fog computing to build Digital Twins is not an entirely new idea. However, the studies surveyed presented mainly abstract concepts, establishing the overall idea with notions on what to expect from a cloud-based Digital Twin deployment, but without realistic implementations. An actual implementation and extensive experimentation would allow a direct analysis of how systems like this can benefit from the use of cloud, fog, and edge interchangeably, and how this could affect the performance of this network of systems.

The three layers edge-fog-cloud architecture has been presented in multiple investigations [27, 28], but none of which presents this architecture in the context of real-time systems or analyzing how much each layer could add or reduce latency. Another problem present in this situation is the scenario of a public cloud/fog: the network resources are shared between heterogeneous systems. Systems in this network require resource competition management, defining process priority for real-time systems to execute their operations without being frozen because other systems have access to the network resources. The modification of existing systems to support Digital Twins implies the addition of a new real-time approach, which would share resources of a network that could already be near its computational limits.

Non-shared cloud and fog Digital Twin network scenarios presented in other works do not accurately describe all real situations, because they demonstrate scenarios where companies would build an entire infrastructure with the sole purpose of supporting Digital Twins. Real situations where they would be implemented most likely already have a working network where the Digital Twin would be inserted, requiring minimal physical change. Cloud and fog, by definition, enable reusable and cheap computation by handling multiple different and independent devices. Networks with a single purpose do not make sense because idle

resources would be wasted, going against the whole concept of cloud computing.

These problems imply two situations: 1) the usage of cloud, fog, and edge computing in the context of workload variability; and 2) the definition of priority for different devices in the network to allow priority computation to be allocated to time-critical systems. Sharing resources in a network requires the definition of a priority policy between devices, to allocate the necessary computation to Digital Twins or other systems. In light of this landscape, the problem addressed in this paper is the investigation of how appropriate a cloud-fog architecture is to support real-time Digital Twins.

#### 4.2 Digital Twin Design Overview

Modern systems built with Digital Twins in mind can implement business logic, ranging from simple computing software, only storing and processing data, to complex systems that can “think” and provide advanced insight for its users. The use of artificial intelligence techniques in all types of tasks is the next step in digitalization and process automation, and the use of cloud-based technologies is an essential tool for building scalable systems for any domain.

For Digital Twins to accurately simulate, analyze, and predict events and situations, they must collect a variety of data from the physical mirrored device or system and process it effectively and timely. Digital Twins must support a collection of different models to accurately describe devices in their full life-cycle phases, which may include simulation and predictive models. As so, Digital Twins require to be equipped with a collection of services to effectively monitor and simulate the physical world it mirrors and computes relevant decisions.

The architecture we designed and implemented is an open-source<sup>[1]</sup> software built as a collection of individual microservices, available under the GNU General Public License v3.0. For the experiments performed in this work, the architecture contains microservices for the client devices, simulating the physical parts, a Digital Twin instance, which is the digital counterpart, and a message broker to intermediate the communication. This work focuses on the communication aspect of the Digital Twin and will not detail internal processing services. These elements are the minimum components required for the Digital Twin communication, emulating the physical and digital parts, from the model proposed by Damjanovic-Behrendt[32].

To execute the experiments, the client devices were simulated with a custom program that sends sequentially numbered messages, with a configurable payload

and publication interval, to an MQTT broker, which uses a topic-based publish/subscribe pattern for communication. This allows decoupling from publisher and subscriber, allowing for each service to run independently and with higher scalability. Each client device publishes their messages with a specific topic known by the subscribed twin instance, identifying their publications as being from that specific client. Numbered messages allow us to analyze the timing aspects of each message, from its publication by the client until it is processed by the twin instance.

The Digital Twin instance is a prototype of the digital part of the system, which implements the required mechanisms for acquisition, storage, and processing of data obtained from its physical counterpart, and is where the physical product is linked throughout its entire life-cycle [3]. The Digital Twin instance is the final destination for the client messages after they are distributed by the broker. In this work, messages sent in the opposite direction, *i.e.* from the twin instance to real devices, were not experimented with and require analysis by future work.

The current implementation uses the Eclipse Mosquitto message broker, version 1.6.10, as it is an open-source message broker implementation of the MQTT protocol, purposely built to be open, simple, and light weight, characteristics heavily associated with IoT contexts [33]. Eclipse Mosquitto is intended for use in situations where there is a need for lightweight messaging, particularly on constrained devices with limited resources [34]. The broker was configured with its default settings, except for the max message queue limitations which were removed, since the broker would discard all messages received if this limit was reached, only retaining the physical restriction of memory limitations of where the broker is being executed.

To meet real-time requirements, the fact that the Digital Twin components of the architecture we designed and implemented are microservice-based allows for a variety of distribution scenarios. These components can be distributed between cloud, fog, and edge computing layers, depending on the needs of the corresponding physical system and the available network resources.

## 5 Experimental Setup and Methodology

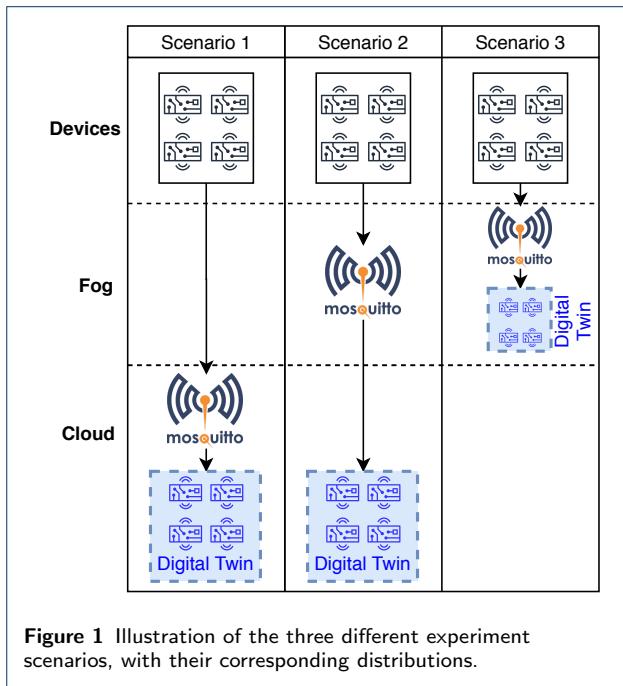
This section describes the performed experiments. It also describes the primary experiment factors, variables that affect the experiment response, which directly impacts the performance of the proposal in the defined scenarios. Factors will be tested in different scenarios, to analyze their impact on the architecture.

<sup>[1]</sup>Source code available at <https://github.com/Open-Digital-Twin>.

### 5.1 Experiment Scenarios

To perform the proposed study, three different experiment scenarios are proposed. These three scenarios are illustrated in Figure 1 and detailed as follows:

- *Scenario 1 - Cloud only:* Processing is done exclusively on a single cloud node. The MQTT broker, where the client publishes and the twin instance subscribes to a data source, resides on this cloud node.
- *Scenario 2 - Fog and Cloud:* The MQTT broker is located on a fog node, closer to the edge, while the twin instance resides on the cloud.
- *Scenario 3 - Fog only:* Both broker and the twin instance resides on a fog node.



### 5.2 Experiment Factors

- *Number of data sources:* twin elements can have multiple data sources, in the form of multiple individual devices, which by themselves can have multiple sensors. For simplification of the problem in this paper, an element is defined as one general part or device of the physical system, which has only one data source. To emulate multiple data sources, it is possible to increase the number of elements accordingly. Each source then sends a certain amount of messages to the broker.
- *Message payload size:* the number of bytes sent in the MQTT message payload generated by the client devices, which will affect bandwidth usage and processing time for the message broker and the twin instance.

- *Source message frequency:* the estimated time between messages sent by the data sources, or client devices, depending on the types of sensors that define the physical system.

- *Transmission latency:* time for the message to travel in two scenarios: 1) from the client to the broker; and 2) from the broker to the twin instance. This latency is directly affected by the distribution of the architecture between cloud and fog.

Each scenario was individually tested under three situations: (A) varying the number of messages sent by each data source; (B) varying the frequency of messages transmissions; and (C) varying the payload of the sent messages.

- *Situation A - Source and number of message:* Situation A varied the amount of messages sent by each data source, in sets of 100, 1000, 10000, and 100000 messages by each source. To test concurrency, the number of data sources was set to 1, 3, and 5, which would add up to a maximum of 500000 messages to be handled by the broker and the Digital Twin instance. The delay between when each message was sent by the data sources was fixed to 10ms, with a payload of 64 bytes.
- *Situation B - Message frequency:* Situation B varied the wait interval between each message sent by the data source, in sets of 10, 20, 40, 80, 160, 320, 640, 1280, and 2560 milliseconds. In this situation, a total of 1000 messages were sent by only one source, each with a payload of 64 bytes.
- *Situation C - Message payload size:* Situation C tests variations of payload sizes on the messages sent by the data source. The payloads were set to 8, 16, 32, 64, 128, 256, and 512 bytes. Each test sent a total of 1000 messages from one source, with a fixed delay of 80ms between each message.

### 5.3 Experiment Hardware Specifications

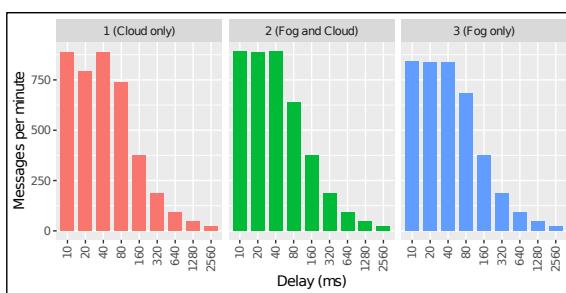
A server running an Intel Xeon E5-2420 (with 6 cores and 12 threads) and 32 GB of RAM was used as the experiment environment. The experiments were executed on three individual Kernel-based Virtual Machines simulating real devices, each named according to their use on the defined scenarios: Cloud, Fog, and Client. The Cloud virtual machine runs with two virtual CPUs and 4GB of memory. The Fog and Client are by definition devices with more constrained resource capacities, so they only have one virtual CPU each, with 1GB of memory.

Regarding network limitations, the only limit inserted was a simulated network latency in the communication between the machines, varying according to each scenario, injected by the usage of *tc-netem* [35].

In scenario 1, a delay of 100ms, with a 10ms variation, was added in the Client. In scenarios 2 and 3, two different delays were added: 40ms, varying 10ms, in the Client, and 20ms, varying 5ms, in the Fog. Delays were injected into the scenarios to simulate the different delays when devices communicate with fog and cloud components, not only with fog having lower hardware capabilities but also having a lower delay. Considering just two machines, without any communication delay difference, the one with more computational power would have a faster response, but computing in the edge/fog layers is the concept that communication can be the bottleneck in how specific distributed systems function.

## 6 Results

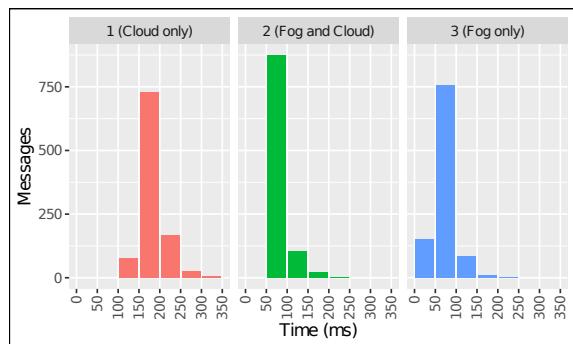
Bringing processing closer to the edge, to handle message exchange, reduces latency for the application. Digital Twins, being composed with a multitude of distributed sensors, could experience high latency and poor connectivity. Proposing processing being done closer to edge is a possible strategy, since being purely dependent on the cloud may not satisfy time-sensitive requirements [5]. The processing unit can receive the data from the source devices and compute its suggestion to the physical system with a shorter response time.



**Figure 2** Messages per minute received by the Digital Twin instance, in each scenario, in the message frequency situation.

An important observation about how the MQTT broker throttles messages was found when testing Situation B, varying the message frequency, in the described scenarios. As evidenced by Figure 2, which shows the amount of messages received per minute in each scenario, varying the delay between publications, the broker has a limit on the number of messages it can receive and publish to its subscribers in the order of 800 messages per minute. We did not set a limit to the message queue in order to prevent the broker from dropping messages under high load situations. Thus, this message per minute rate is a hard limit set by the broker in order to prioritize receiving publications

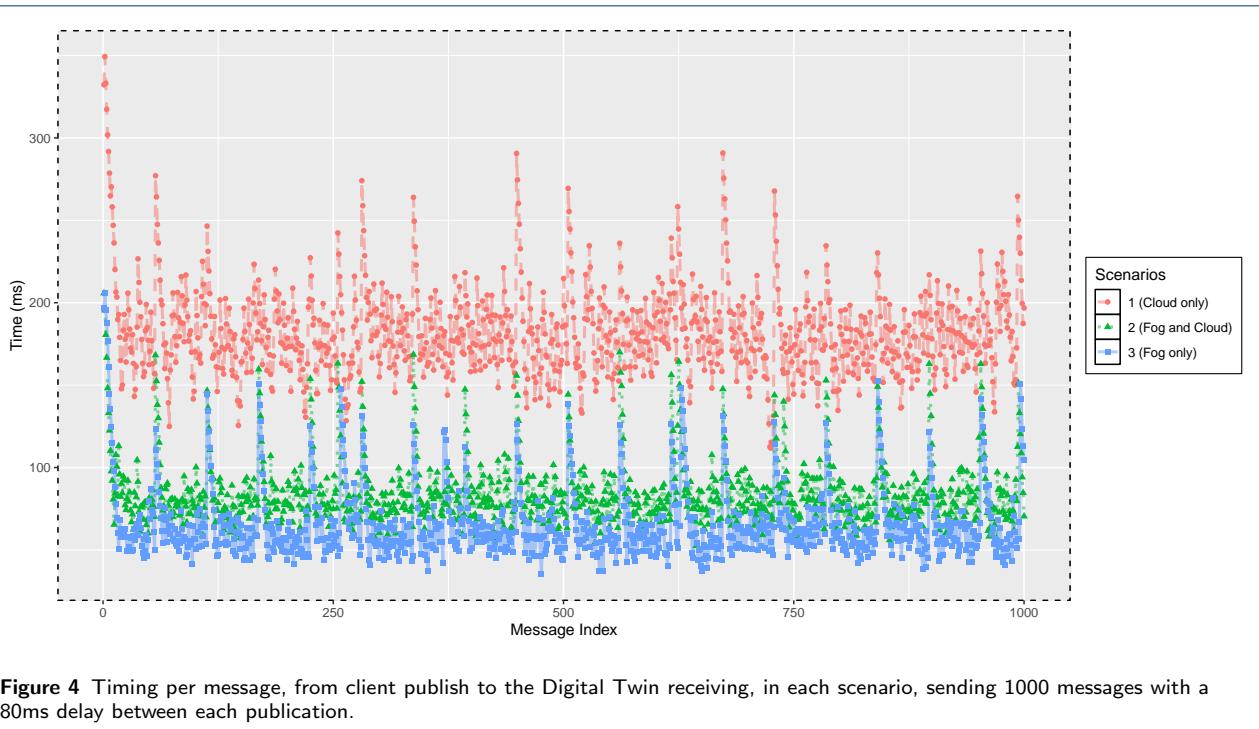
and to not overload subscribers through limiting the number of in-flight messages. Eclipse Mosquitto can be configured to increase this in-flight message limitation or even remove it. However, doing so could cause message ordering to break or it could overload subscribers and should be tested in a case by case scenario, according to the number of messages sent by the real device.



**Figure 3** Timing histogram of 1000 messages, from client publish until reaching the Digital Twin, in each scenario.

Figure 3 presents a timing histogram of 1000 messages in the three scenarios, on situation C with a payload of 64 bytes and 80ms interval between each message, from the time it is published by the client until received by the Digital Twin instance (subscriber). Results from this experiment show not just the average timing of messages, but the time of each individual publication. Scenario 1 presents messages with a higher time compared to the two other scenarios, where most messages are received in the 50 to 100 ms interval, in this experiment. Considering a real-time system with a communication requirement of 200ms in the Cloud only scenario (1), 805 of the 1000 messages would respect this limitation, representing 80.5% of the total messages sent by the client. Meanwhile, only one message would not arrive in time for this imposed system limitation in the Fog and Cloud (2) and Fog only (3) scenarios, respectively. Further reducing this limitation to 150ms, 92.4% of messages sent in scenario 1 would be lost, while 0.21% and 0.09% would not respect the requirement by scenarios 2 and 3.

More benefits depend on the architecture scale of implementation in the fog/cloud levels. By having resources closer to the edge, it is possible to perform pre-processing as soon as possible, before centralizing the data in the cloud. Figure 4 makes this clear by showing the time each message takes from being sent by the client until received by the twin instance in each deployment scenario. The situation shown in this image sends 1000 messages, each one with a delay between



each publication of 80ms. Table 1 provides a summation of the average times of this experiment, providing a general overview of all the published messages.

**Table 1** Average times per message, in milliseconds, from client to twin, for the 80ms delay experiment situation.

Scenario	Average	Std Deviation	Max	Min
1	182.45	27.932	349.35	111.82
2	84.43	20.417	205.83	52.51
3	66.08	22.626	205.63	35.13

The usage of cloud computing enables devices to store data in a centralized location, but large amounts of data being transmitted, natural to the concept of Digital Twins with a large scale network of sensors, implies an increasing delay, affecting response times from the server.

As shown in Table 1, deploying to the Fog and Cloud scenario (2) and the Fog only scenario (3) reduced average times per message by 54% and 64% respectively, compared to the Cloud only scenario (1). Maximum times were also reduced by 41% for scenarios 2 and 3, as were the minimum times by 53% and 69%.

As a response from the results presented in this work, fog computing is shown as a viable alternative to reduce the effects of this problem, reducing the transmission delay in order to match real-time requirements. Further reductions depend on improving lower layers of the network components, via extra connectivity and reduced latency in general.

## 7 Conclusion and future work

This paper investigated how the usage of a cloud-fog architecture for Digital Twins can help in meeting real-time requirements of these systems.

The tested architecture includes a client, for sending sensor data, the MQTT broker, for handling messages sent by the client, and the Digital Twin instance, which is subscribed by the broker. Tests were executed in three different scenarios, changing the deployment locations of the architectural elements where the broker and twin reside in the cloud, where the broker resides in a fog node, and the twin on the cloud, and lastly where both the broker and twin share the same fog node.

The experiments tested these different scenarios on how they would handle the number of messages, different frequency of messages sent, and payload size. From these experiments, it was concluded that the distribution of these Digital Twin elements closer to the edge reduce communication delays for the end application, allowing faster response by bringing preprocessing into distributed locations. This particular distribution allows the Digital Twin to meet real-time systems timing specifications.

Having all the required data flow from the client devices, the Digital Twin can run application-specific algorithms handling this input and be able to suggest actions to the physical systems respecting their real-time requirements.

As future work, some limitations could be addressed, as the problem regarding the real-time priority policy for the MQTT broker: message handling should prioritize real-time subscribers. The MQTT protocol has no definition of priority for subscriptions, but a broker-defined configuration indicating priority topics could be used, dedicating more resources to attempt a best-effort approach into meeting real-time requirements.

#### Abbreviations

IoT: Internet of Things; MQTT: Message Queuing Telemetry Transport.

#### Availability of data and materials

Please contact authors for data requests.

#### Competing interests

The authors declare that they have no competing interests.

#### Funding

This work was supported in part by the Brazilian National Council for Scientific and Technological Development (CNPq) and in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Finance Code 001. We also thank the funding of CNPq, Research Productivity Scholarship grants ref. 313893/2018-7, ref. 306614/2017-0, ref. 309505/2020-8 and ref. 420109/2018-8.

#### Acknowledgements

Not applicable.

#### Authors' contributions

F. Knebel collaborated with the problem formalization and methodology, participated in the software implementation and resulting data investigation. J. Wickboldt collaborated with the problem formalization and methodology, participated in the software implementation and resulting data investigation. E. de Freitas collaborated with the problem formalization and methodology, and the result investigation. All authors wrote, read, and approved the final manuscript.

#### Author details

Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, BR.

#### References

1. Grieves, M.: Origins of the Digital Twin Concept. Technical report, Florida Institute of Technology / NASA (2016)
2. Kopetz, H.: Real-time Systems: Design Principles for Distributed Embedded Applications. Springer, Wien, AT (2011)
3. Grieves, M., Vickers, J.: Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In: Kahnen, F.-J., Flumerfelt, S., Alves, A. (eds.) Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches, 1st edn., pp. 85–113. Springer, Rochester, USA (2017)
4. Schneider, D., Trapp, M.: B-space: dynamic management and assurance of open systems of systems. *Journal of Internet Services and Applications* **9**(1), 1–16 (2018)
5. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* **98**, 289–330 (2019)
6. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. *IEEE Internet of Things Journal* **3**(5), 637–646 (2016)
7. Khan, W., Rehman, M., Zangoti, H., Afzal, M., Armi, N., Salah, K.: Industrial internet of things: Recent advances, enabling technologies and open challenges. *Computers & Electrical Engineering* **81**, 106522 (2020)
8. Wan, J., Chen, B., Wang, S., Xia, M., Li, D., Liu, C.: Fog computing for energy-aware load balancing and scheduling in smart factory. *IEEE Transactions on Industrial Informatics* **14**(10), 4548–4556 (2018)
9. Verma, P., Sood, S.K.: Fog assisted-IoT enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal* **5**(3), 1789–1796 (2018)
10. Tang, B., Chen, Z., Hefferman, G., Wei, T., He, H., Yang, Q.: A hierarchical distributed fog computing architecture for big data analysis in smart cities. In: Proceedings of the ASE BigData & SocialInformatics 2015, pp. 1–6 (2015)
11. Perera, C., Qin, Y., Estrella, J.C., Reiff-Marganiec, S., Vasilakos, A.V.: Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys (CSUR)* **50**(3), 1–43 (2017)
12. Nikoloudakis, Y., Markakis, E., Mastorakis, G., Pallis, E., Skianis, C.: An NFV-powered emergency system for smart enhanced living environments. In: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 258–263 (2017). IEEE
13. Li, C., Xue, Y., Wang, J., Zhang, W., Li, T.: Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Computing Surveys (CSUR)* **51**(2), 1–34 (2018)
14. Dizdarević, J., Carpio, F., Jukan, A., Masip-Bruin, X.: A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Comput. Surv.* **51**(6) (2019)
15. Voell, C., Chatterjee, P., Rauch, A., Golovatchev, J.: How Digital Twins Enable the Next Level of PLM – A Guide for the Concept and the Implementation in the Internet of Everything Era. In: Product Lifecycle Management to Support Industry 4.0, pp. 238–249. Springer, Cham, CH (2018)
16. Stojanovic, N., Milenovic, D.: Data-driven Digital Twin approach for process optimization: an industry use case. In: 2018 IEEE International Conference on Big Data (2018)
17. Cardin, O.: Classification of cyber-physical production systems applications: Proposition of an analysis framework. *Computers in Industry* **104**, 11–21 (2019). doi:[10.1016/j.compind.2018.10.002](https://doi.org/10.1016/j.compind.2018.10.002)
18. Mell, P., Grance, T., et al.: The NIST definition of cloud computing. National Institute of Science and Technology, Special Publication, 800 **145** (2011)
19. Marinescu, D.C.: Cloud Computing: Theory and Practice. Morgan Kaufmann, Waltham, USA (2017)
20. Deng, J., Huang, S.C.-H., Han, Y.S., Deng, J.H.: Fault-tolerant and reliable computation in cloud computing. In: 2010 IEEE Globecom Workshops, pp. 1601–1605 (2010). IEEE
21. Tordera, E.M., Masip-Bruin, X., Garcia-Alminana, J., Jukan, A., Ren, G.-J., Zhu, J., Farré, J.: What is a fog node? a tutorial on current concepts towards a common definition. arXiv preprint arXiv:1611.09193 (2016)
22. Reyna, A., Martín, C., Chen, J., Soler, E., Díaz, M.: On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems* **88**, 173–190 (2018)
23. Chen, Y., Deng, S., Ma, H., Yin, J.: Deploying data-intensive applications with multiple services components on edge. *Mobile Networks and Applications*, 1–16 (2019)
24. Atlam, H.F., Walters, R.J., Wills, G.B.: Fog Computing and the Internet of Things: A Review. *Big Data and Cognitive Computing* **2**(2), 10 (2018)
25. Ray, P.P., Dash, D., De, D.: Edge computing for Internet of Things: A survey, e-healthcare case study and future direction. *Journal of Network and Computer Applications* **140**, 1–22 (2019)
26. Aazam, M., Zeadally, S., Harras, K.A.: Offloading in fog computing for iot: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems* **87**, 278–289 (2018)
27. Qi, Q., Zhao, D., Liao, T.W., Tao, F.: Modeling of Cyber-Physical Systems and Digital Twin Based on Edge Computing, Fog Computing and Cloud Computing Towards Smart Manufacturing. In: ASME 2018 13th International Manufacturing Science and Engineering Conference (2018). American Society of Mechanical Engineers Digital Collection
28. Kim, T.: Cyber Physical Systems Framework of Edge-, Fog-, and Cloud-Computing. In: Proceedings of the International Conference on Embedded Systems, Cyber-physical Systems, and Applications (ESCS), pp. 10–14 (2019). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)
29. Mohan, N., Kangasharju, J.: Edge-Fog cloud: A distributed cloud for Internet of Things computations. In: 2016 Cloudification of the Internet of Things (CIoT), pp. 1–6 (2016). IEEE

30. Ahmad, S., Afzal, M.M.: Deployment of Fog and Edge Computing in IoT for Cyber-Physical Infrastructures in the 5G Era. In: International Conference on Sustainable Communication Networks and Application, pp. 351–359 (2019). Springer
31. Fernández-Caramés, T.M., Fraga-Lamas, P., Suárez-Albela, M., Díaz-Bouza, M.A.: A Fog Computing Based Cyber-Physical System for the Automation of Pipe-Related Tasks in the Industry 4.0 Shipyard. *Sensors* **18**(6), 1961 (2018)
32. Damjanovic-Behrendt, V., Behrendt, W.: An open source approach to the design and implementation of digital twins for smart manufacturing. *International Journal of Computer Integrated Manufacturing* **32**(4-5), 366–384 (2019)
33. Banks, A., Briggs, E., Borgendale, K., Gupta, R.: MQTT Version 5.0. Accessed: 2020-08-09 (2019).  
<https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>
34. Light, R.A.: Mosquitto: server and client implementation of the MQTT protocol. *Journal of Open Source Software* **2**(13), 265 (2017)
35. Hemminger, S., Ludovici, F., Pfeifer, H.P.: NetEm - Network Emulator. Accessed: 2020-08-22 (2011).  
<https://man7.org/linux/man-pages/man8/tc-netem.8.html>