Computer Standards & Interfaces 81 (2022) 103600

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Facilitating the monitoring and management of structural health in civil infrastructures with an Edge/Fog/Cloud architecture

Cristian Martín [*], Daniel Garrido, Luis Llopis, Bartolomé Rubio, Manuel Díaz

ITIS Software, University of Malaga, Arquitecto Francisco Peñalosa, 18, Málaga 29071, Spain

ARTICLE INFO

Keywords:
Structural health monitoring
Internet of things
Cloud computing
Fog computing
Edge computing
Lightweight virtualization

ABSTRACT

Structural Health Monitoring (SHM) is nowadays a requirement for civil infrastructures like tunnels, bridges and viaducts. With the advances provided by the Internet of Things (IoT) in recent years in areas such as wireless communications, miniaturisation and application protocols, this paradigm is spreading to the wider society to ensure appropriate protection of critical infrastructures. Both fog and edge computing have also strongly contributed to SHM and the IoT by bringing computing as close as possible to where data is produced, thereby reducing the latency response with respect to traditional cloud integrations. In this paper, an Edge/Fog/Cloud architecture for SHM in civil infrastructures is presented. The main goal of this architecture is to create and provide a flexible framework involving all the required components for the management, monitoring and deployment of SHM solutions, enabling high availability and easy distribution of the components over the architecture. The architecture has been evaluated as an alternative to the real deployment in a tunnel of a cloud architecture showing the benefits of adopting an Edge/Fog/Cloud hierarchy.

1. Introduction

Some civil infrastructures (CI) related to transport, hydrological management or the generation and distribution of energy play a strategic role in the development of many activities essential for human beings. Failures or malfunction of these infrastructures are able to seriously affect these essential activities. Early detection of faults in CIs keeps maintenance and repair costs lower than if they are detected at a future stage when the problem has worsened. In this sense, Structural Health Monitoring (SHM) as presented in our previous work [1] plays a decisive role in fulfilling this purpose. SHM is a non-destructive technique for evaluating the state of a structure based on its dynamic response, which enables detecting, locating and quantifying possible damage.

The most advanced structural monitoring systems are based on Wireless Sensor Networks (WSN), which are one of the most important components in the Internet of Things (IoT) paradigm [2]. These networks provide real-time monitoring of the infrastructure at a low cost. Additionally, data collection can be continuous and artificial intelligence can be applied to predict their structural health and even to predict the CI life cycle. A typical architecture based on WSN is composed of a set of monitoring nodes including sensors and communication modules. Data from these nodes are collected in a gateway, which is responsible for sending data to the cloud. Cloud platforms contribute not only to infrastructure monitoring but also to the application of analytical methods to detect damage to the infrastructure and for the prognosis of a failure. The combination of WSN and cloud architectures has facilitated the integration of the IoT concept in the context of SHM for CI.

Cloud architecture provides access to computation, storage and even connectivity with easy access. However, these centralised architectures can create delays and performance issues for devices and data that are far away from a centralised public cloud or data center source. If the analysis of the data is centralised in the cloud e.g., to analyse emergency situations, any communication failure or delays in the response will limit the operation and will affect human lives.

The architectures based on fog and edge computing represent promising alternatives that complement cloud-based systems, especially for a rapid response to emergency situations. Fog computing is a computing paradigm introduced for the purpose of extending the cloud capabilities (computation, storage and network services) closer to the edge of the network [3]. Generally speaking, it is a geographically

* Corresponding author.
E-mail addresses: cmf@lcc.uma.es (C. Martín), dgarrido@lcc.uma.es (D. Garrido), luisll@lcc.uma.es (L. Llopis), tolo@lcc.uma.es (B. Rubio), mdr@lcc.uma.es (M. Díaz).

https://doi.org/10.1016/j.csi.2021.103600
Received 24 November 2020; Received in revised form 30 September 2021; Accepted 30 October 2021
Available online 3 November 2021

distributed computing architecture connected to multiple heterogeneous resource-limited devices (network devices, mini data centers, lightweight servers) that forms a bridge between the cloud and the edge of the network to facilitate the deployment of new IoT applications. The edge computing concept is interchangeable with fog computing [4]. Possibly, the key difference is the location inside the IoT network where the processing of data is performed. In the case of fog computing, the data is processed as close as possible to the IoT devices, while edge computing pushes the limits even further by allowing IoT devices and connected gateways to process some data locally. These two paradigms reduce the latency and the bandwidth in the communications between the IoT and cloud systems, which are highly time-sensitive in SHM. More recently [5], these architectures have been accompanied with lightweight virtualisation technologies like containers (e.g., Docker), which have enabled a lightweight way of scaling and reallocating components, applications and services. Due to their lightweight nature, they can be installed in a wide range of systems including embedded devices like a Raspberry Pi [6]. Consequently, these technologies enable fault tolerance through orchestration systems (e.g., Kubernetes) and lightweight horizontal and vertical migrations [7] in fog/edge architectures, which are needed to meet the time requirements in mission-critical systems and to balance the system load when required.

Despite the advances provided by fog/edge architectures and lightweight virtualization technologies to mission-critical applications, their management and deployment still require a lot of manual processes. A challenge in these mission-critical environments is how to facilitate the management, monitoring and configuration in a versatile and simple way. In this paper, an SHM Edge/Fog/Cloud architecture for CI is defined where the target environments are easy to install and extremely versatile, allowing users to design, install and read data in any type of infrastructure. The main goal of this architecture is to facilitate, in a flexible and very versatile way, the monitoring, configuration and management of the complex CI deployments with the benefits of a fog/edge architecture. The whole process of an SHM deployment is addressed by this architecture, from the monitoring nodes to the infrastructure management until the data analysis for the damage detection. All these steps are provided in a common interface that helps and reduces maintenance operations for administrators and final users in their daily tasks on civil infrastructures. This work has been validated in a civil infrastructure for damage detection.

The main contributions of this approach to improve control over civil infrastructures are:

- We propose a hierarchical (edge-fog-cloud) architecture to manage structural health monitoring applications.
- The architecture enables users to largely customise and improve their deployments and monitoring solutions.
- Each layer (edge, fog, cloud) has its own function with the overarching goal of reducing the latency response for critical situations (damage detection).
- The hierarchy is scalable from the cloud, fog and edge perspectives with the benefits in terms of latency response and bandwidth brought by adopting this architecture.
- The architecture has been validated in a real Tunnel use case.

Our proposal uses a novel combination of up-to-date technologies. To the best of our knowledge, this combination has been used for the first time in the domain of Structural Health Monitoring. Next, we highlight the motivation for using these technologies and how they help to improve our architecture.

- Internet of Things. Traditional SHM technologies used monolithic ad-hoc architectures that cannot be reused. Moreover, many of the components of a solution (hardware or software) could not be replaced in an easy way. By using IoT technologies and the proposed solution we obtain more flexible solutions where components can be easily replaced, deployed and configured by final users.
- Edge-Fog-Cloud Computing. This is one of the stronger points of our proposal. As shown in this paper, the use of such hierarchical architecture allows taking several benefits. Mainly, latency and bandwidth usage are reduced, which are required features for mission-critical applications like SHM. Other related advantages are scalability, dynamic deployment and stronger fault tolerance.
- Lightweight Virtualization. The use of software containers provides several advantages to our architecture. First of all, it allows easy deployment of the analysis techniques. These containers include all the required dependencies. As a result, smaller self-contained pieces of software can be distributed. Second, the use of containers through orchestrators such as Docker Swarm, allow efficient management of computational resources including CPU, memory or storage. Finally, containers can be easily redeployed or moved where they are more convenient for the tasks they have to do (i.e. along with the fog or cloud).

The rest of the paper is organised as follows. In Section 2 related work is discussed. Section 3 presents the proposed SHM architecture and all of its components. In Section 4 we describe a real use case scenario where the SHM architecture has been validated. An evaluation of the architecture is shown in Section 5. Finally, Section 6 discusses the conclusions and future work.

2. Related work

There seems to be a broad consensus [8], [9], [10] in regard to the limitations of the Cloud when considering the huge data volume generated by IoT applications. Primarily, the bandwidth limitation and high latencies are considered the two most important problems related to the IoT-cloud relationship. Most of these approaches present fog as a complementary alternative to cloud providing solutions to these problems where the fog nodes can work, storing information, providing computational resources and communicating only a subset of processed data to the cloud. Our work follows the same approach. As the evaluation results show, latency and bandwidth are reduced in a fog architecture, which is very important for scenarios like CI monitoring.

To the best of our knowledge, only a few works have proposed a fog/edge architecture to reduce the execution latency in SHM. In [11], a similar approach as this work is presented. A framework for SHM with fog computing that transmits data to the cloud with 5G networks is presented. This work presents a novel solution for power saving and maximising the lifetime of the WSNs by the definition of cluster heads in the IoT devices for sending data to the Fog. Our architecture provides all the necessary components to facilitate the management, monitoring and deployment of SHM solutions that can be easily moved thanks to its containerisation. Moreover, that work has been evaluated only theoretically whereas the framework proposed in this paper has been validated in a civil infrastructure. In [12], the authors demonstrated a promising performance and opportunity of adopting LoRA and a Fog computing architecture in places where there is no availability of internet connection in a smart health monitoring system. In this paper, the proposed solution can also be applied to environments with limited connectivity like the Tunnel deployment used for validation. Atmosphere [13] presents a context and situational-aware collaborative IoT architecture based on three-tiers, complex event processing and agent-oriented software. Two-way communications are allowed among the three tiers of the architecture, which is different to our proposal where our edge-fog-cloud levels communicate only from lower to upper levels more focused on reducing latency and delegate computation to upper levels.

In [14] an interesting survey on state-of-the-art IoT literature is presented. In this survey, the authors investigate enabling technologies, services and open research issues related to the Cloud-to-Things continuum. They propose an architectural framework based on Fog/Edge computing-based IoT (FECIoT), which considers load balancing, resilience, fault tolerance, data sharing and reduction in the Cloud-to-Things communication. Most of these features are present in our system. The authors also mention vertical and horizontal communication within the IoT system. In our approach, the use of containers helps cover this feature. Finally, they present several architectural styles with several layers. In this sense, our approach follows a four-layer architecture (sensing, network, service and application) where the service layer includes data analysis and the application layer provides the user interface.

The use of microservices in IoT is explored in [15]. In this approach, the authors use *linked-microservices* as a way of distributing the computation across different computing nodes in the IoT architecture. Using this approach, they try to reduce the latency and bandwidth of IoT applications. They explore four different architectures namely cloud, fog, hybrid and fog+cloud with an evaluation of several machine learning algorithms and different types of datasets. They conclude that service decomposition reduces the data consumption by 10% - 70% depending on the architecture, algorithm and dataset. In our approach, we use containers as a way of service decomposition. In our containers, we execute different algorithms in a similar way to that proposed in that paper.

Fault tolerance is very important when considering SHM, where computer security in terms of availability and integrity is mandatory. The work presented in [16] presents a survey of fault-tolerant techniques based on redundancy for the IoT. They distinguish three different areas in an IoT system: sensing, routing and control. The first two areas have fairly developed techniques. However, the control area is less developed and is based on state-machine replication with consensus protocols. Fault tolerance in our approach is based on replication techniques using containers. Master nodes are responsible for the monitoring of worker nodes. In the case when worker node fails, the manager can deploy new instances to maintain availability and integrity. Several of the experiments also show how we can increase the number of managers with a low penalty, which contributes to enforcing the QoS of our system.

The work presented in [17] has some similarities with ours. The authors use Principal Component Analysis (PCA) as a data reduction method for SHM in the monitoring of a viaduct. However, they have to address the memory and computing limitations of embedded low-cost gateways. In their work, they use a memory-efficient implementation of the streaming History PCA algorithm. They obtain good compression factors together with a significant memory reduction. The final signal can be reconstructed with a low degradation. The work in [6] also has some similarities with ours. The authors use containers, and an open-source cloud solution (OpenStack) and Raspberry Pi clusters like our approach. They aim to find a Platform as a Service (PassS) solution that can be used for service packaging and orchestration. Our solution is more focused on SHM in civil infrastructures. Our contribution provides an Edge/Fog/Cloud infrastructure with the aim of reducing the latency and bandwidth of analysis techniques in SHM deployments. Furthermore, it is intended to facilitate the management and monitoring of SHM deployments, for which a system and a Web management user interface have been provided so that administrators can easily manage all the SHM components of civil infrastructures like tunnels. Work presented in [18] is more focused on unattended WSN networks. They present the concept of Deploy&Forget network where WSN can be deployed in an easy way to ensure unattended and long-lasting operation. However, different levels of edge-fog-cloud are not considered and they are out of scope in this work.

Deep learning techniques applied to fog and cloud are explored in [19]. The authors present EdgeLens, a framework that is able to work in two different modes (High-Accuracy mode or Low-latency mode) depending on whether data obtained from the sensors is compressed or not. The architectural design of the solution is similar in some aspects to our proposal. They have input sensors, gateways (collecting information from the sensors) and *Aneka* nodes deployed as containers in the Fog or the Cloud (responsible for executing deep learning algorithms). An important aspect is how the models are trained. In this case, the training of the models is done separately on high-performance computers. Some parameters such as accuracy, response time, network bandwidth or power consumption are studied in this paper. Both works use different analysis techniques: deep learning versus modal analysis, which can also be complementary [20]. An advantage of our approach can be the absence of the training phase, which is not required in our case. On the other hand, deep learning techniques can be more flexible to changes in the system.

Random Forest techniques for prediction are used in [8]. This paper presents a conceptual framework for the IoT. The authors recognise the importance of IoT data analytics and present several limitations of only-cloud approaches (e.g. bandwidth and high latency). As an alternative, they propose an Analytics Everywhere framework composed by a network of tasks using edge, fog and cloud resources. This framework is composed by three different component types: resource capability (computing power), analytical capability (analytical tasks) and data life-cycle (raw, aggregated or filtered data). The most difficult task is to determine how to map different analytical capabilities with the most appropriate resource capability based on a data-life-cycle of an IoT application. Our approach is different in several aspects: we leverage containers capacity for auto-scaling, fault tolerance and replication. This work defines its own network of tasks, resources or analytical tasks. We think our approach is more interoperable in the sense we use a widely used technology based on Docker containers. Second difference is related to the purpose of this work. They are more focused on obtaining insights from the IoT. Instead, we want to detect defects on infrastructures.

Clemente et al. [9] present a Distributed Cooperative Data Analytics (DCDA) middleware for the IoT. DCDA enables a high-level vision of the system at the edge. DCDA considers three different levels of processing and analytics. On the low level, they control actions that require immediate attention such as alerts and notifications. On the medium level, they are able to generate analytics information using historical data. Finally, the high level is more oriented to decision-making processes. Nodes can work in two different modes: *Task sharing mode*, where several nodes work together to solve a specific problem, and *Cooperation mode*, where a node uses resources from other nodes. Fault tolerance, scalability and energy consumption are also studied in this proposal. DCDA has been applied to several seismic use cases that require a real-time response in a similar situation to our use case. This work also shares with our proposal the division of processing at different levels. However, it is different in the sense they use every level for a different purpose. The lowest level is used for alerts and notifications. The intermediate level is associated with real-time analytics with data visualization and the higher level is devoted to decision-making processes. They also use a middleware that is equivalent to our container-based approach.

3. Structural health monitoring architecture

The main goal of the SHM architecture is to facilitate the monitoring and management processes involved in the deployment of civil infrastructures. Fig. 1 shows an overview of this SHM architecture. At the bottom the edge nodes obtain the information from the monitoring nodes and upload it to the Container Infrastructure, which has been provided over a Cloud/Fog scheme, which does not only enable the easy portability and scalability of the system when required, but also reduces the latency response and bandwidth of the communications between the IoT and the cloud, which are desirable aspects in applications that require a low latency and generate large amounts of data, like mission-critical applications.

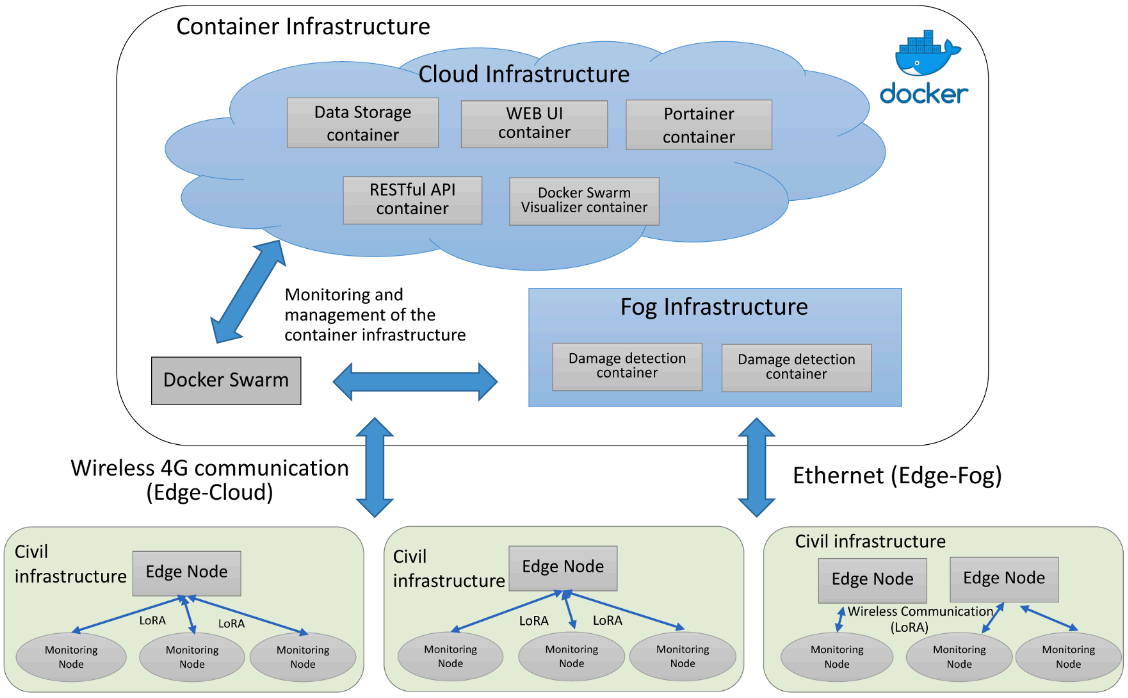


Fig. 1. Structural Health Monitoring Architecture.

3.1. *Monitoring nodes*

The IoT part of the architecture comprises a self-install kit for Structural Health Monitoring of Civil Infrastructures. It is easy to install and versatile, allowing users to design, install and read data in any type of infrastructure. Monitoring nodes are the hardware components which integrate the necessary sensors such as accelerometers and magnetometers to monitor the CI. The LoRA wireless communication technology [21] has been adopted in the nodes for its low power consumption (they can run on batteries for years), long-range (they can be deployed in large critical infrastructures) and low interference. LoRA is a type of low-power wide-area network (LPWAN) designed to allow long-range communications at a low kbit/s rate between things.

The flexibility of monitoring nodes allows users to select the sensors to integrate into the node to gather the necessary data to analyse the structural health in the infrastructure. The final users by means of a Web application can decide on which sensors to use. In fact, each node can include different sensors depending on the place in the CI where it is to be installed. In Fig. 2 a prototype of the monitoring node used in this architecture is shown. The hardware used for the monitoring nodes (pointed in Fig. 2 in the same order as below) is as follows:

1. SD Card Module. To store the measurements in case of no connection to the edge node.
2. Lora Module RN2483 16311AD connected to a Xbee explorer.
3. Magnetometer MAG3110.
4. RTC DS1307. To establish a time in the monitoring nodes.
5. Camera ArduCam for tunnel photogrammetry. Not used in this work.
6. Accelerometers ADXL362 (2 axis) and ADXL345 (3 axis). Mainly used in this work for tunnel monitoring analysis.
7. Temperature and humidity DHT22 sensor.
8. Moteino Mega. It is the brain of the monitoring node, chosen for its low power consumption (not seen in the Figure).

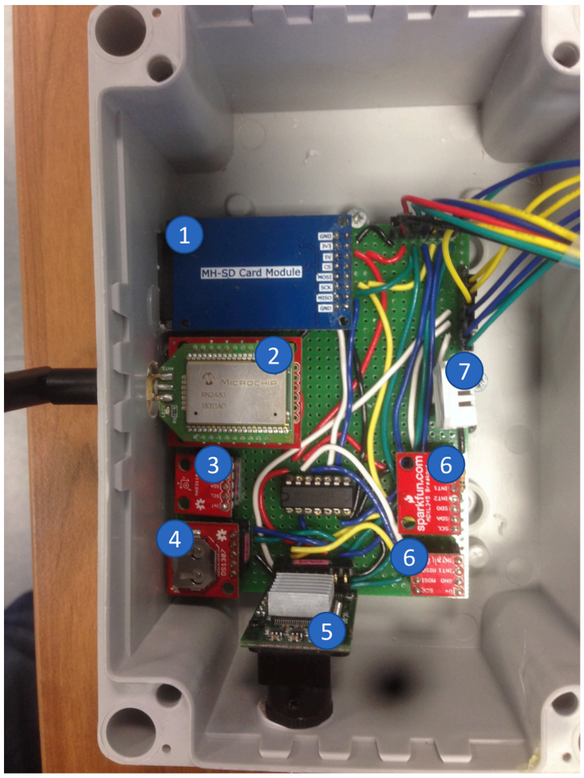


Fig. 2. A prototype monitoring node with a set of sensors for monitoring a civil infrastructure including a three-axis accelerometer, temperature and humidity sensors and a Arducam.

3.2. *Edge nodes*

Edge nodes are the devices that are placed at the border of the network, which enable the connection between the Container Infrastructure and the monitoring nodes. The edge devices are also responsible for receiving and filtering the sensor measurements from the monitoring nodes as configured in the Monitoring and Management Web UI. For instance, once a CI has been created in the Web UI by the final users, a monitoring configuration can be designed. This configuration includes the desired measurements to be monitored from the monitoring nodes and their monitoring frequency, among other configurations. Once the configuration has been confirmed by the users, the corresponding edge devices receive all the required information to enable that monitoring. From that moment on, the edge devices send requests to the corresponding monitoring nodes to configure their monitoring as defined in the Web UI and the edge devices start receiving the measurements in the times configured in the system. The edge devices provide two means of communication: 1) LoRA for both the incoming and outgoing connection with the monitoring nodes; and 2) the communication with the Container Infrastructure through 4G (for the connection with the cloud) and Ethernet (for the connection with the Fog). Therefore, the edge devices are the LoRA head nodes in the communication with the monitoring nodes. The edge devices in this architecture are usually deployed in a Raspberry Pi. Monitoring nodes are synchronised with the edge devices to minimise time synchronisation errors in damage detection [22]. The communication between the monitoring nodes and the edge nodes does not use a standard IoT protocol like CoAP or MQTT, but rather a proprietary protocol has been used. However, one of those standard protocols will be considered as future work to improve the interoperability between the monitoring nodes and the edge devices.

To avoid communication interruptions and losing any measurements received from the monitoring nodes, all the measurements received in the edge devices are stored in a circular local file system. Then, a periodic task is in charge of processing the stored measurements and uploading them to the Container Infrastructure when the communication is available. Bear in mind that the edge devices are also resource-constrained devices and for large deployments, replication is required to communicate with the monitoring nodes.

3.3. Container infrastructure

Monolithic software designs do not allow the scalability of the architecture, nor managing different levels of load at run-time. For that reason, all the components that comprise the architecture have been incorporated as microservices inside a container infrastructure in order to enable both scalability, high availability and vertical and horizontal migrations. Container virtualisation technologies have garnered a lot of attention in the last few years due to their features: fast processes of building containers, high density of services per container and high isolation between instances [23]. In contrast to traditional hypervisors, lightweight virtualisation technologies implement the virtualisation of processes through containers in the operating system. This reduces the overhead of the hardware and virtual device virtualisation in traditional hypervisors, permitting the deployment of a high density of containers.

Therefore, the use of this virtualisation technology can be applied in resource-constrained devices, such as edge gateways. Docker[1] is the most representative example of a container platform and it has been adopted in this architecture as the container infrastructure. Another reason to choose Docker is that it is also well-supported by all the infrastructure deployed in this work, e.g., the portable fog cluster. The following components have been incorporated and distributed inside the Docker containers comprising the software architecture:

- **Monitoring and Management Web UI**. This component serves the management and monitoring Web UI that will be used by the registered users in the system. This Web UI has two roles: the administrator, who is responsible for managing all the deployments, the registered users and the system configuration; and the local user, who only has access to monitor and manage his/her own deployment.
- **RESTful API**. The RESTful API is used both by the Web UI to serve the user's information and by the edge nodes to upload the measurements from the civil infrastructures. Therefore, this component represents both the data source and data sink of the system. This inter-operable API can also be integrated with third-party systems that are intended to be integrated within the system [24].
- **Data storage**. The data storage is responsible for storing all the information involved in the architecture. This information includes the measurements received in the RESTful API, the registered users in the system and the defined configuration for the system's deployments.
- **Damage detection analysis**. This component is responsible for executing the data analysis tasks that will process the measurements received from the civil infrastructures for damage detection.
- **Docker Swarm Visualizer**. Docker Swarm Visualizer is an open-source project that provides a user-friendly Web UI for visualising the nodes belonging to a Docker cluster and the containers deployed on them.
- **Portainer**. The Docker-project Portainer[2] has been used to manage the Docker cluster and their Docker resources (containers, images, volumes, networks and more). Portainer is a management Web UI with a community edition that allows the Docker cluster to be easily managed without having to write multiples lines of script code.

Docker allows the deployment and management of containers. However, other suitable characteristics in container systems such as orchestration and clustering are not provided in Docker itself. These are provided by another Docker related project, Docker Swarm[3]. Docker Swarm enables the orchestration of containers, providing high availability and load balancing on them through container replicas and monitoring. Docker Swarm monitors each node of the system (cloud and fog) and allocates and distributes the containers based on the resource utilisation and the availability of the nodes. The configuration of the containers, such as the number of replicas and the port-forwarding to the host, is defined in a configuration file, known as the compose file. This file is used by Docker Swarm to obtain the container infrastructure configuration and needs in order to distribute the containers throughout the cluster. Therefore, Docker has been adopted as the container platform and Docker Swarm as the orchestration platform in the SHM architecture. To facilitate the visualisation of the containers in the cluster, we have used Docker Swarm Visualizer. Finally, the tool Portainer has been adopted for the management of the Swarm cluster and the containers. Both Docker Swarm Visualizer and Portainer are single lightweight Docker containers, therefore they can be easily deployed as services like the rest of the components of the container infrastructure.

3.4. Deployment of the container infrastructure in the fog and the cloud

Fog and cloud are both suitable candidates for allocating the container infrastructure. In fact, it could solely be deployed in the cloud along with its benefits of global access, scaling and high availability. However, as discussed in the Evaluation, the Fog provides a better performance for the timely microservices of the SHM system. For this reason, the damage detection analysis, which is the most critical task since it allows the identification of damages on civil infrastructures is executed in the Fog. The rest of the components such as the monitoring and management Web UI, the data storage and the RESTful API are less critical and are placed initially in the cloud. Nevertheless, thanks to the continuum monitoring and orchestration of the container infrastructure provided by Docker Swarm, these components can be easily relocated (downgraded or upgraded) at run-time depending on the current state and load of the infrastructure (fog, cloud). In this scenario, the edge

[1] Docker: https://www.docker.com/

[2] Portainer: https://www.portainer.io/

[3] Docker Swarm: https://docs.docker.com/engine/swarm/

nodes send the needed measurements for the damage detection to the fog, whereas other measurements that are stored and not needed in the analysis such as temperature and humidity are directly sent to the cloud. Every service is localised by its hostname since the networking is transparently managed by Docker Swarm, thereby in the case of service displacement along the infrastructure this localisation is transparent to the edge nodes.

On the other hand, this infrastructure could also be entirely deployed in Fog, and this can be required in the cases where there is no Internet connection and a portable Fog is used. However, the benefits that bring the combination of both paradigms in terms of adjusting the whole infrastructure to the current needs of the system and the reduction of latency provided by the Fog deserve the utilisation of both paradigms for mission-critical applications.

3.5. Monitoring and management web UI

The Monitoring and Management Web UI provides an accessible, simple and unified interface to the final users so that they can manage and visualise the SHM deployments of the system. One of the first actions to operate with the system is the creation of a new civil infrastructure deployment. This is a simple step where users define the location of the deployment, its address and related information.

Over the deployments, users can define the monitoring of the infrastructures. Once users have defined the basic information of the monitoring such as access to the power grid (to be taken into account in the installation of the monitoring nodes), users allowed, total monitoring distance and pedestrian access, they can select the number of monitoring nodes and their sensors to be deployed by specialised technicians in the civil infrastructure as shown in Fig. 3. This interface is one of the possible interfaces offered by the Management Web UI and allows administrator users to configure the basic information about the monitoring of the infrastructure such as the monitoring name, the number of nodes, and type of sensors, as described before. This is the first interface that administrators face when setting up new monitoring of a critical infrastructure. Next, they can configure the number of monitoring intervals and the time of the day when the monitoring nodes should obtain the configured measurements.

Once the civil infrastructure deployment and monitoring configurations have been defined, a token per monitoring is generated in the system, which has to be included in the edge devices configuration. Tokens enable edge devices to upload the monitoring data and establish a connection with the Container Infrastructure in a secure way.

Finally, when the monitoring nodes have been installed by specialised technicians and configured through the edge devices and the system, the SHM architecture starts receiving monitoring data at the time defined, which can be visualised in Web UI as shown in Fig. 4. Users can filter the monitoring data selecting the measure to visualise and the date interval.

The whole process for configuring an SHM in a civil infrastructure from when it is registered until the system starts receiving monitoring data is summarised in Fig 5. Through this interface administrators and end users can easily configure all the steps involved in an SHM deployment in a civil infrastructure.

3.6. Damage detection of the civil infrastructures

To evaluate the real-time status of the civil infrastructures, a modal analysis of the monitoring data received is performed in the architecture. Modal analysis refers to the study of the inherent dynamic properties of engineering infrastructures in the frequency domain [25]. Modal analysis is used to formulate a mathematical model of dynamic behaviours and is very important to determine the status of the infrastructures.

Natural frequencies and mode shapes are two of the modal parameters extracted from the modal analysis. Natural frequencies are the frequencies at which an infrastructure tends to oscillate in the absence of damping and driving forces. Natural frequencies are commonly used to determine the properties of an infrastructure at the design phase. For instance, it is important to design infrastructures that do not match the frequency of expected earthquakes in a region, otherwise, both frequencies can join and amplify and the infrastructure can experience structural damage. On the other hand, mode shapes are the patterns of motion of these infrastructures at the natural frequencies. For the modal analysis, the acceleration data (3-axis sensor) is used. In particular, we used the operational modal analysis (OMA), applied successfully in our previous work for high-speed railway infrastructure monitoring [26]. The analysis performs the frequency domain decomposition (FDD)[27], where the relation between the tunnel monitoring data $x(t)$ and the measured response $y(t)$ can be expressed as shown in formula (1).

$$S_{yy}(\omega) = H^{*}(\omega)S_{xx}(\omega)H^{T}(\omega) \tag{1}$$

where $S_{yy}(\omega)$ is a matrix (r x r) of the power spectral density (PSD) of the response, and r the number of time series; $H(\omega)$ is a matrix (m x r) of frequency response function (FRF) of the system, and superindices T and * indicate respectively the transposed matrix and conjugated complex matrix; and $S_{xx}(\omega)$ is a matrix (r x r) of the PSD of the input.

The FRF matrix can be expressed in the form of residuals and poles as shown in formula (2).

$$H(\omega) = \sum_{k=1}^{n} \frac{R_k}{\omega - \lambda_k} \frac{R_k^{*}}{\omega - \lambda_k^{*}} \tag{2}$$

where n are the number of models; λ_k is the pole; and R_k is the residue. R_k can be expressed as $R_k = \phi_k \gamma_k^T$, where ϕ_k and γ_k are the mode shape vector and the modal participation vector, respectively

Through an orthogonal decomposition, the spectral eigenvectors and eigenvalues of the system are obtained through the spectral density of the response, as shown in formula (3).

$$S_{yy}(\omega) = \sum_{k=1}^{M} \varphi_k(\omega)\theta_k(\omega)\varphi_k^{*T}(\omega) \tag{3}$$

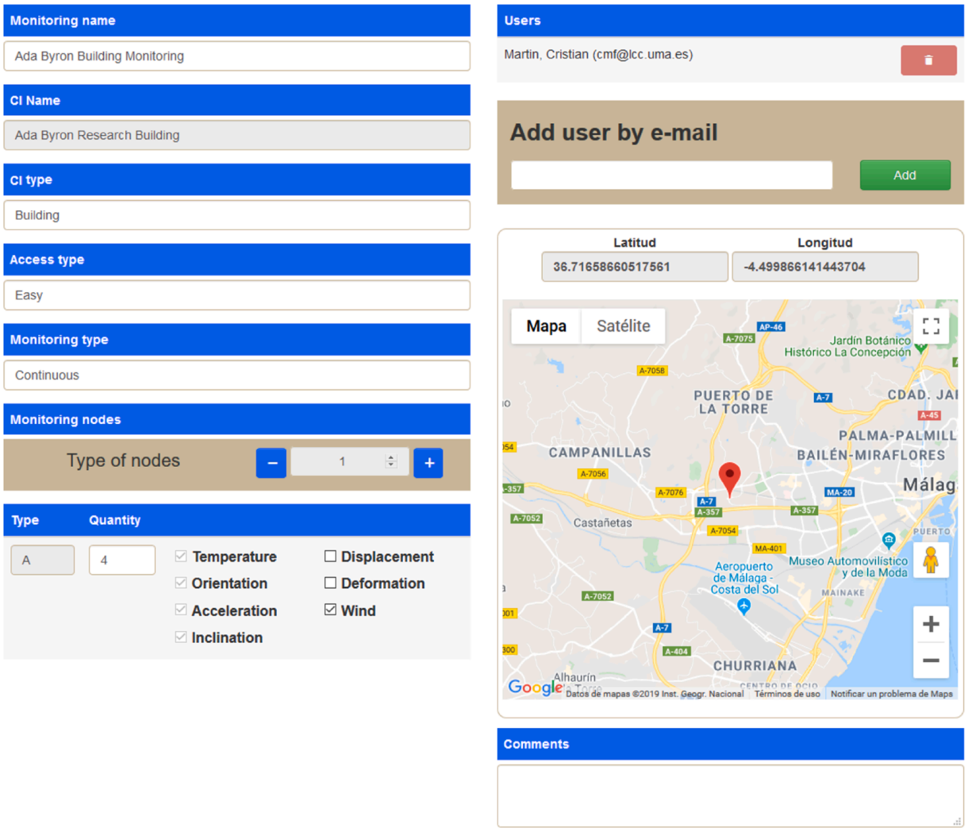
where M is the number of decompositions; $\varphi(\omega)$ are the spectral eigenvectors of the system; and $\theta(\omega)$ are the spectral eigenvalues of the system.

Near a peak, the eigenvectors and eigenvalues of the first order are dominant in the response of the system in terms of energy; and with these values can be obtained the natural frequencies and mode shapes. The ith mode shape (m_i) associated with the ith natural frequency (ω_i) can be estimated using the first-order eigenvector: $m_i = \varphi_{i1}$.

The procedure carried out for the OMA analysis is as follows:

1. The acceleration and sampling frequency from the tunnel measured by the monitoring nodes are obtained through the edge devices. Measurements are filtered (Butterworth second-order low-pass and high pass filters) and sent by the edge devices to the Container infrastructure.
2. Next, in the Container infrastructure, for each axis, the PSD is calculated and then combined into the cross-spectral density (CSD) matrix using the Welch calculation method. The CSD calculates the energy distribution through two synchronised time series of the same structure but with different locations of the sensors.
3. The decomposition of the CSD matrix is performed to obtain the eigenvectors and eigenvalues of the system.
4. The natural frequencies and associated mode shapes will be obtained through the eigenvalues and eigenvectors of the first order decomposed.
5. Finally, natural frequencies and mode shapes are checked for any deviation.

Damage and even its location in civil infrastructures can be detected



Monitoring name	Ada Byron Building Monitoring
CI Name	Ada Byron Research Building
CI type	Building
Access type	Easy
Monitoring type	Continuous
Monitoring nodes	Type of nodes 1

Type	Quantity		
A	4	☑ Temperature	☐ Displacement
		☑ Orientation	☐ Deformation
		☑ Acceleration	☑ Wind
		☑ Inclination	

Users

Martin, Cristian (cmf@lcc.uma.es)

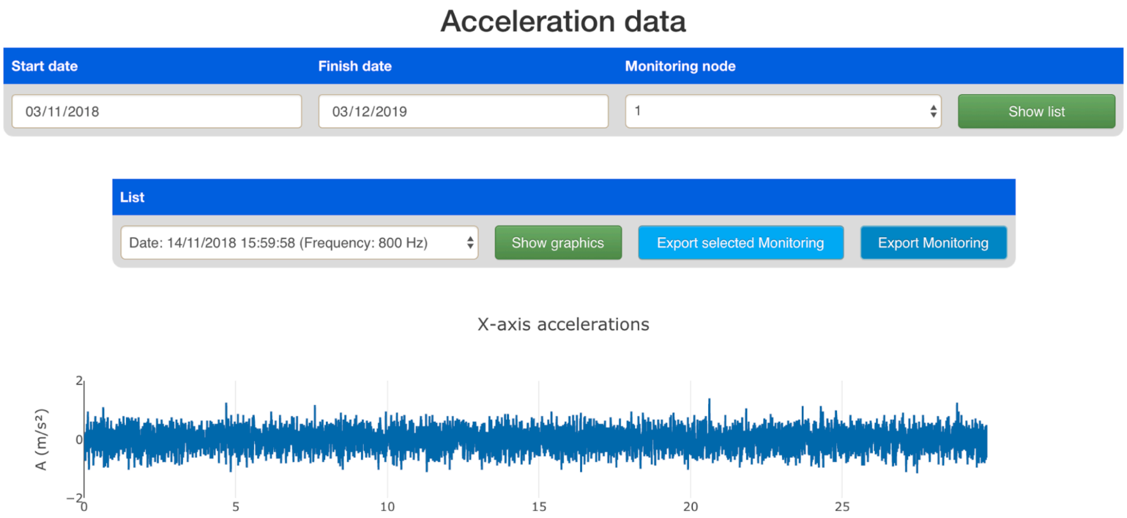
Add user by e-mail

Latitud	Longitud
36.71658660517561	-4.499866141443704

Comments

Add
Mapa
Satélite
PUERTO DE LA TORRE
CAMPANILLAS
BAILÉN-MIRAFLORES
PALMA-PALMILLA
Málag
Castañetas
Aeropuerto de Málaga - Costa del Sol
Museo Automovilístico y de la Moda
Jardín Botánico Histórico La Concepción
CHURRIANA
Alhaurín
Google
Datos de mapas ©2019 Inst. Geogr. Nacional Términos de uso Notificar un problema de Maps

Fig. 3. Monitoring configuration of a civil infrastructure deployment.



Acceleration data

Start date	Finish date	Monitoring node
03/11/2018	03/12/2019	1

List

Date: 14/11/2018 15:59:58 (Frequency: 800 Hz)

Fig. 4. Monitoring data received from a monitoring node in a civil infrastructure deployment.

by monitoring the changes and deviations in the natural frequencies and mode shapes [28]. Therefore, continuous monitoring of the dynamic characteristics of civil infrastructures is necessary to determine their structural health. In the work presented here, modal parameters are identified by the peak-picking method. However, other methods like curve-fitting through the Least-Squares Complex Frequency Domain (LSCF) estimator from the Python open-source modal analysis software OpenModal [29] can be used. Curve fitting is the process of matching a mathematical expression to a set of empirical data points.

In this work, the modal analysis is performed to detect damages in the infrastructure. By real-time monitoring the civil infrastructures any alteration will be detected in a very short of time. In the case of damage detection, an alarm will be generated, which can be visualised by the administrators and end users in the Monitoring and Management Web UI.

Thanks to the use of a fog infrastructure, this analysis, which can take a considerable amount of time and produce a large amount of data, can be distributed across, over or throughout the Fog to reduce the response time and actuate as soon as possible in the case of changes in the modal parameters.

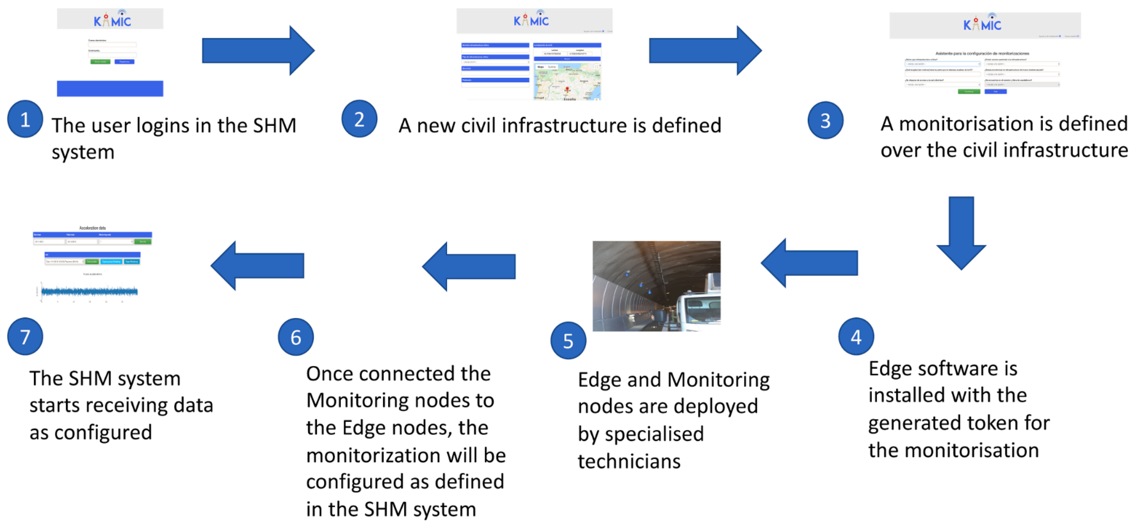

1 The user logins in the SHM system
2 A new civil infrastructure is defined
3 A monitorisation is defined over the civil infrastructure
4 Edge software is installed with the generated token for the monitorisation
5 Edge and Monitoring nodes are deployed by specialised technicians
6 Once connected the Monitoring nodes to the Edge nodes, the monitorization will be configured as defined in the SHM system
7 The SHM system starts receiving data as configured

Fig. 5. The process of configuring an SHM in a civil infrastructure.

4. Use case: An Edge/Fog/Cloud architecture for SHM of a tunnel

The architecture has been evaluated as an alternative to a cloud-based SHM system of a tunnel. This system is a real case located in the southeast of Spain where a structural health monitoring analysis was necessary. Three monitoring nodes were registered in the civil infrastructure and installed on one side of the tunnel as shown in Fig. 6. The monitoring nodes had integrated some sensors such as magnetometer, temperature and humidity sensors and 3-axis accelerometers. The strength, temperature and humidity measurements are stored in the system and can be visualised in the Web UI. However, the most significant measurements are the accelerations since the modal analysis uses them to detect alterations in the structural health of the infrastructures. The accelerations were configured in the Web UI to be collected every ten minutes for ten seconds with a frequency of 800Hz. An edge node with a 4G connection was deployed in the tunnel to serve the monitoring nodes and connect them with the cloud. A portable fog was also deployed in the tunnel and connected through Ethernet with the edge device. The fog infrastructure was deployed during the deployment of the monitoring nodes and edge devices in the tunnel and was used for the evaluation of this work. During installation, one lane of the tunnel was closed and traffic was reduced. After the deployment and evaluation, the fog infrastructure was uninstalled due to the lack of permissions and space in the tunnel to position it. As future work, we intend to study the feasibility of integrating a portable fog infrastructure like this one inside the tunnel itself to have it permanently. The data size for each data collection was around 2,9MB, which is a considerable amount of data per monitoring node to evaluate the advantages and disadvantages of the edge/fog schema when compared with the traditional cloud-based scenario.

5. Evaluation

The evaluation aims to measure the performance of the SHM architecture in two different scenarios. On the one hand, the container infrastructure has been deployed in a commercial cloud platform. This is a traditional scenario where the main limitations in the IoT are supplied by cloud computing: processing power, storage and networking at the expense of an increase of latency response and bandwidth. In this scenario, the edge nodes are still placed in the architecture and continuously send the measurements to the Google Cloud platform as configured in the system. The connection of the edge nodes to the Google platform is through a 4G connection. On the other hand, we have a portable fog computing infrastructure placed between the edge devices and the cloud platform, which is intended to reduce the latency between the IoT and the cloud platform in the modal analysis. This portable fog infrastructure was connected through Ethernet with the edge devices in the tunnel. The fog computing infrastructure comprises a portable cluster of Raspberry Pi model 3 (1GB RAM, 64GB micro-SD) connected through a Cisco Catalyst 2960 10/100 switch. Raspberry Pi is an embedded device that supports Docker containers [6] and allows us to have a 24-node fog computing infrastructure at a low price (around 2000 EUR in total). The Raspberry Pi cluster is shown in Fig. 7 and

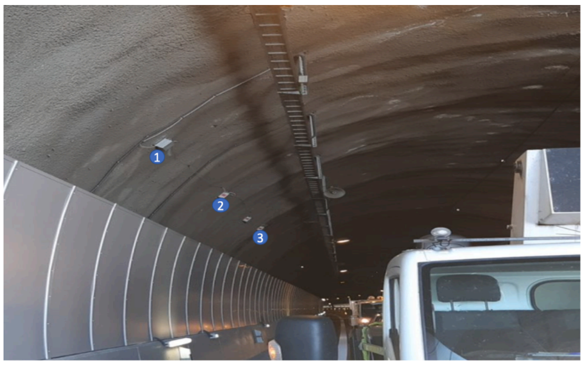


Fig. 6. Monitoring nodes installed in a Tunnel deployment of the architecture in Spain.

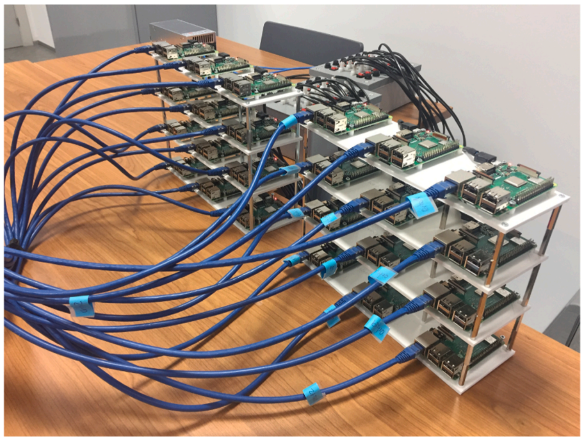


Fig. 7. Raspberry Pi cluster used as Fog infrastructure in the SHM architecture.

known as FogPi [30]. Each layer of the cluster and the grey boxes to distribute the power have been printed with a 3D printer. We have also used HypriotOS[4], which is a minimal Debian-based operating system for Raspberry Pi, optimised and designed to run Docker containers.

Since we only could deploy 3 monitoring nodes and one edge device, to stress the Fog/Cloud architecture, we collected the data gathered by the edge device from the monitoring nodes on the Tunnel (around 2,9MB per monitoring node) to execute up to 64 simultaneous clients (edge devices) sending data to the fog infrastructure. The fog infrastructure was deployed in the Tunnel and the gathered data from the edge device was sent concurrently by a PC (where all the clients were executed) to provide a higher data ingestion in the architecture. Thanks to this, we have up to 64 edge simulated devices for further stressing of the architecture, otherwise we would only have one edge device deployed in the tunnel which would generate a lower load. Therefore, edge devices are simulated through a PC during the evaluation in the Tunnel to stress the architecture. The infrastructure for the evaluation thus comprises a PC, our FogPi infrastructure and Google Cloud. Generally, even though there could be large CIs, in an SHM deployment there is usually a single edge node. Thus this evaluation aims to evaluate the performance of the architecture from one to multiple SHM deployments. We developed a Python script to simulate the behaviour of the edge nodes sending data to be processed by the modal analysis in the fog/cloud. This script creates one thread per client sending data and was executed on a Windows 10 PC with 16GB of RAM. This Python script also measures the latency response and throughput of analysis performed in the fog/cloud from the moment the measurement is sent until the response is obtained. All evaluations were carried out 100 times taking the average values. Both in the cloud and the fog, the modal analysis was deployed inside Docker containers through Docker Swarm and Portainer.

The operational modal analysis described in Section 3.6 has a time and space complexity marked by the frequency domain decomposition performed [31]. The solution proposed based on Edge/Fog/Cloud computing in this paper does not alter the complexity of this approach since each analysis component is executed in an isolated way through Docker containers, and replication only provides high availability, load-balancing and fault tolerance.

5.1. Latency response and throughput

In the first evaluation, we have measured the latency response and the throughput in the Fog and the Cloud by multiple simultaneous edge nodes sending a full monitoring from the tunnel deployment (around 2,9MB). Bear in mind that the connection between the simulated edge nodes and the cloud platform is provided through a 4G connection (the one we adopted in our previous deployments), and the connection between the fog and the edge nodes is through Ethernet. We consider that our fog cluster can be portable to civil infrastructures in order to have wired connection with the edge nodes.

To have equitable conditions both in the Fog and the Cloud, one replica of the modal analysis with 1GB of RAM has been deployed for this evaluation. Fig. 8 shows the latency response in both scenarios. The results demonstrate that with one edge node the latency response is 2,87 times higher in Google cloud, whereas with 32 nodes the latency is 4,4 times higher. This is due to the network latency since in the cloud scenario all the information has to go through the network whereas in the Fog the computation is closer.

The throughput of this evaluation is shown in Fig. 9. The higher throughput in both scenarios is achieved with low numbers of edge nodes, which is predictable due to the number of connections. Overall, the Fog achieves a higher throughput than the Cloud.

[4] HypriotOS: https://github.com/hypriot

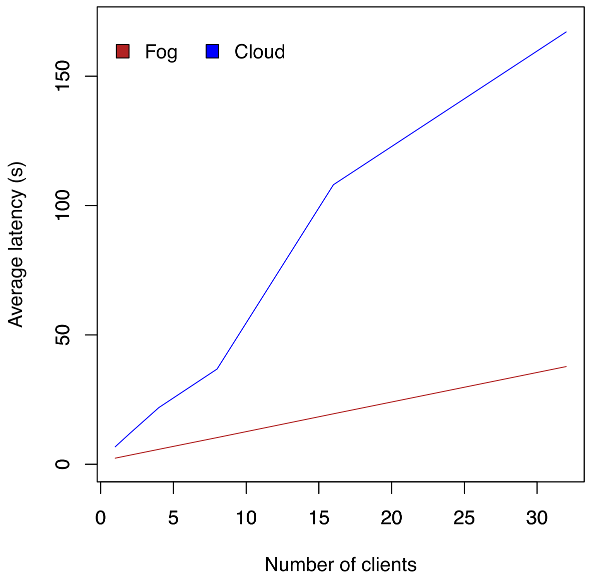


Fig. 8. Average latency response of the modal analysis in the Fog and the Google cloud.

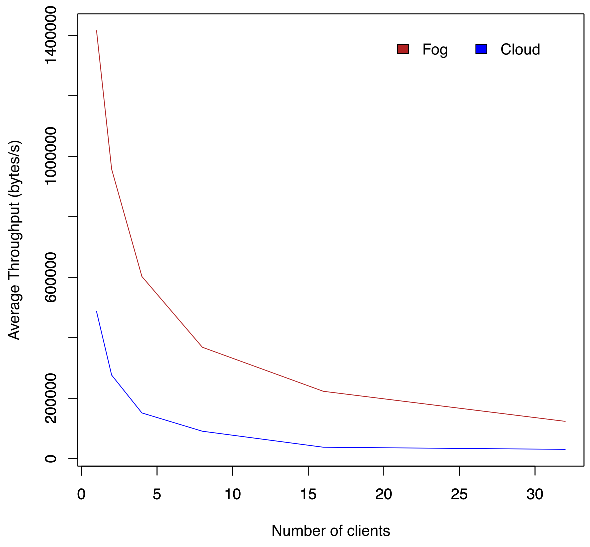


Fig. 9. Average throughput of the modal analysis in the Fog and the Google cloud.

5.2. Data size impact on latency

In order to evaluate the impact of the monitoring data size on the latency response, another evaluation was carried out adjusting the monitoring data size by controlling the frequency. This helps us to decide when the Cloud is more feasible than the Fog. In this evaluation we have also increased the processing power both in the cloud with 8 CPUs and 8GB of RAM (the maximum with our plan); and in the Fog, by having 16 container replicas distributed in the cluster. This is not entirely unfair, as we suppose that the CPU frequency in Google Cloud is much higher than our fog cluster (700 Mhz per Raspberry Pi). Figs. 10, 11, 12 show the latency results of the Fog and Cloud with different monitoring data sizes and 1, 32 and 64 edge nodes respectively. With the increase of the processing power both in the Cloud and the Fog, the system can deal with more edge nodes and the latency response is lower than with 1 replica (Fig 8).

In relation to the monitoring data size, in the three cases the latency response differs in centiseconds with small data sizes (a few of Hz).

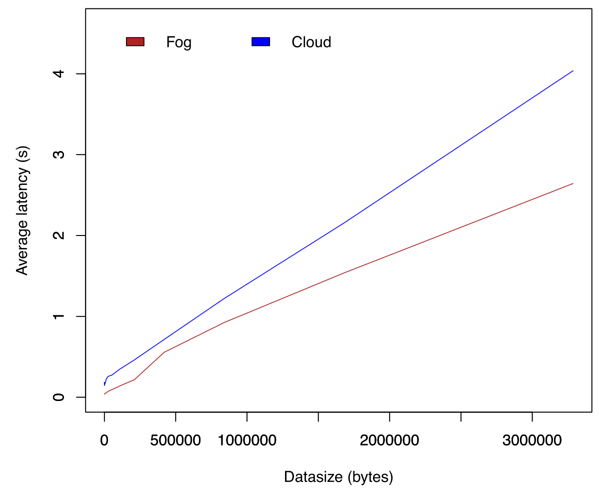


Fig. 10. Latency response with 1 Edge node and different data sizes.

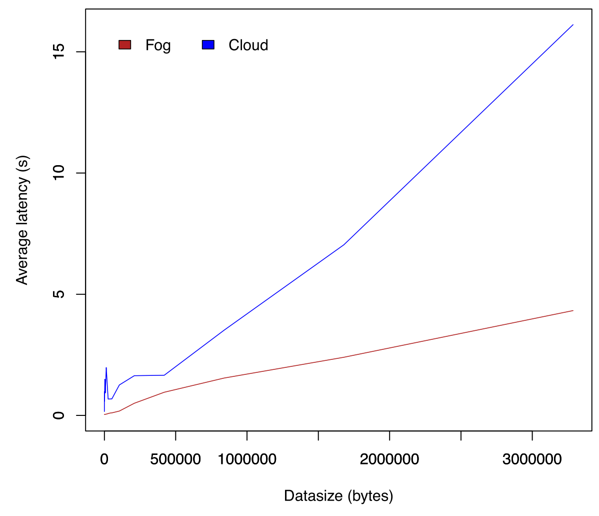


Fig. 11. Latency response with 32 edge nodes and different data sizes.

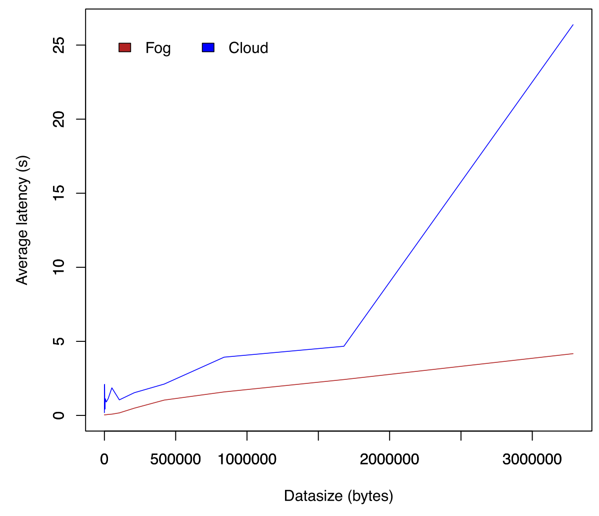


Fig. 12. Latency response with 64 edge nodes and different data sizes.

However, when the monitoring data size increases, especially with large amounts of monitoring nodes, the difference turns to an increase of seconds in the Cloud. We can conclude that with a lightweight monitoring without real-time requirements a cloud platform could be adopted without the need to invest in a fog infrastructure. Nevertheless, in the case of large amounts of monitoring data required for processing units like modal analyses and the necessity of a low latency, the fog provides an alternative to be considered.

5.3. Replication

Thanks to the capabilities offered by the Container Infrastructure in terms of lightweight portability and distribution, the system can scale (due to high amounts of load) and downgrade (to release used resources) when required. Figs. 13 and 14 respectively show the impact of the modal analysis in container replicas to the latency response and throughput in the architecture. The replication is achieved by deploying container replicas of the modal analysis component through the Portainer web UI. The replicas are continuously monitored through Docker Swarm and are deployed on both the portable fog and the cloud platform. It can be seen that with a higher number of replicas, the system can offer a higher quality of service (QoS). On the other hand, with a high number of replicas the system can also have an overhead as shown in Fig 15, which zooms the latency response of the highest replicas. In this architecture and the evaluation performed, the best performance is achieved with 16 replicas.

In a Docker Swarm cluster, the cluster is managed by managers and a group of workers which are continuously monitored are responsible to execute the containers upon request. These managers also manage the worker failures, fault tolerance and the load distribution in the system. A failure in a one-manager cluster can take the cluster down. Therefore, to have a fault-tolerant cluster the managers have to be replicated. Fig. 16 shows the latency response impact of having multiple managers in our Fog cluster. For simplicity, we have performed this evaluation with 16 replicas (the best performance obtained in the latency response). The results denote an increase of latency response (centiseconds) with an increase of up to 4 managers, which is admissible given the higher level of QoS in the architecture.

6. Conclusions and future work

In this paper, an Edge/Fog/Cloud Architecture for Structural Health Monitoring (SHM) in Civil Infrastructures (CI) has been presented. The main goal of this architecture is to facilitate the monitoring, configuration and management of complex CI deployments in a flexible and

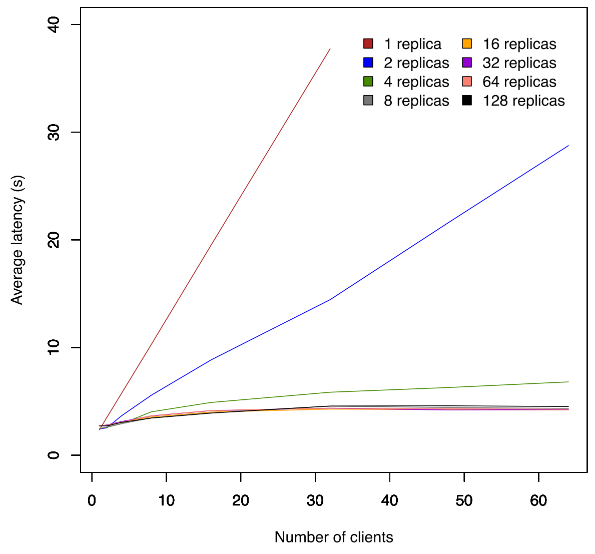


Fig. 13. Average latency response of the modal analysis in the Fog with different numbers of replicas.

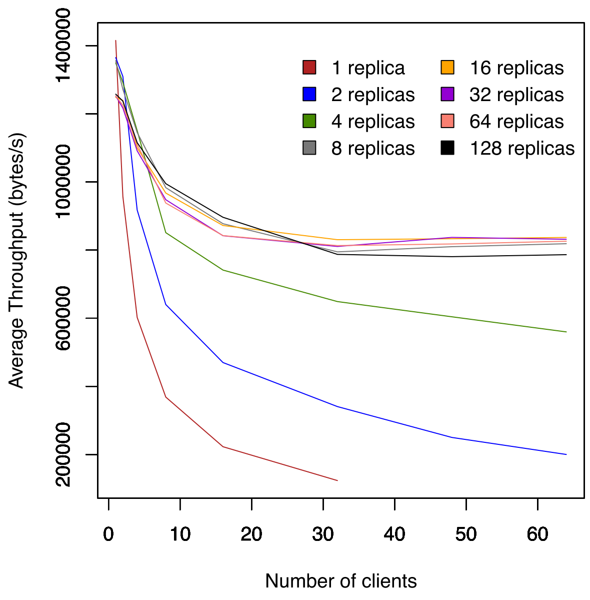


Fig. 14. Average throughput of the modal analysis in the fog with different numbers of replicas.

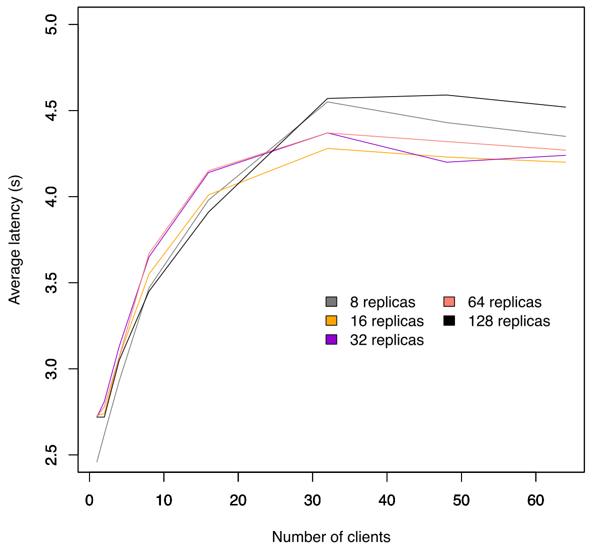


Fig. 15. Average latency response of the modal analysis in the Fog with different numbers of replicas (8–128).

highly versatile way. These steps can be handled in a common interface that helps both administrators and final users in their daily tasks and reduce maintenance operations on CIs. For this reason, this architecture enables the configuration of the IoT solutions together with their sensors and monitoring intervals, the visualisation of the CI information and the data processing to detect the structural health of the infrastructures in a unified interface. To achieve this goal, this architecture aims to cover most of the stack involved in a CI deployment: how the information is obtained from the infrastructures (monitoring nodes); how the information is acquired and sent to the cloud/fog (edge nodes); an infrastructure to easily allocate the components in a Fog/Cloud architecture (Container Infrastructure); the information processing (damage detection of the infrastructures); and a unified interface to manage and monitor the CIs (Monitoring and Management Web UI). State-of-the-art paradigms and solutions such as fog computing and lightweight virtualisation adopted in this architecture not only allow the scalability and portability of the system when required but also reduce the bandwidth

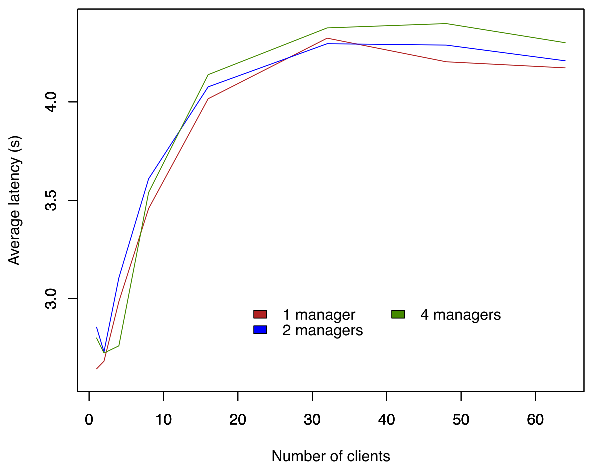


Fig. 16. Average latency response of the modal analysis in the fog with different numbers of managers.

and latency response in IoT communications. In addition, the flexibility of containers simplifies the integration of new analysis models for SHM in the Fog.

This architecture has been evaluated as an alternative to a cloud architecture deployed in a real civil infrastructure, a tunnel in the southeast of Spain where several monitoring nodes are installed. The evaluation performed shows how fog computing (through a cluster of Raspberry Pis) can reduce both the latency and the bandwidth in the communication between the monitoring nodes and the cloud. Moreover, the lower processing time in the modal analysis in the Fog compared to the Cloud is desirable for scenarios like CI where a low latency is required. Finally, this architecture helped administrators in their daily work of management and monitoring of the civil infrastructures. With our proposal, administrators can remotely monitor the CI in real-time using an application, which was previously performed by manual inspection. Moreover, with this system they can easily configure the inspection including node configuration, periodicity of inspections, etc. Finally, it is not required to have a technical IT profile to use the system, which is also an extra motivation.

As for future work, we have several directions planned on the roadmap. On the one hand, an integration of the current advances in LPWAN networks for the communication between the monitoring nodes and the edge nodes is intended. This would enable the use of the wireless technology LoRA with current IoT standards like CoAP, thus facilitating the adoption of this architecture. Interesting alternatives can be based on 6G [32], which can provide edge intelligence with ultra-reliable low latency. On the other hand, this architecture can integrate our previous work in [33] to provide monitoring nodes with fault tolerance, i.e., the architecture would automatically detect service disruptions in the monitoring nodes to adapt the processing and information acquired of the CI with other available data sources deployed in the infrastructures. Even though we can control the deployment of components in the cloud and fog, a mechanism/algorithm that allows the dynamic portability of the container infrastructure over the infrastructure would optimise the architecture load and latency response when required. Streaming techniques [34] are also in our roadmap. When a huge volume of data has to be processed, these techniques can help in data distribution, replication and fault tolerance. Security is also a very important factor. We can try to improve our proposal as outlined in [35].

Our solution is not restricted to SHM, but it is also extendable to other fields where the benefit from a hierarchical architecture in several tiers can be used. For instance, Cloud-based design and collaborative manufacturing [36,37] are also several fields where our approach edge-fog-cloud could be successfully applied.

Finally, in this work we address one of the main steps of an SHM system: the detection of damages. However, other steps [38] such as damage localisation and quantification and the prediction of the remaining useful life of infrastructures are also important in an SHM system. We plan to extend this work by integrating techniques and mechanisms to address these steps in SHM. Thanks to the adoption of containers, these steps can be easily incorporated as new microservices in the infrastructure.

CRediT authorship contribution statement

Cristian Martín: Software, Validation, Writing – original draft. **Daniel Garrido:** Validation, Writing – original draft. **Luis Llopis:** Supervision, Conceptualization, Writing – review & editing, Funding acquisition. **Bartolomé Rubio:** Supervision, Conceptualization, Writing – review & editing, Funding acquisition. **Manuel Díaz:** Supervision, Conceptualization, Writing – review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript.

Acknowledgment

This work is funded by the Spanish projects RT2018-099777-B-100 (“rFOG: Improving Latency and Reliability of Offloaded Computation to the FOG for Critical Services”), PY20_00788 (“IntegraDos: Providing Real-Time Services for the Internet of Things through Cloud Sensor Integration”), and UMA18FEDERJA-215 (``Advanced Monitoring System Based on Deep Learning Services in Fog’’). Funding for open access charge: Universidad de Malaga/CBUA. Cristian Martín was with a postdoc grant from the Spanish project TIC-1572 (”MIsTICa: Critical Infrastructures Monitoring based on Wireless Technologies”).

References

[1] L. Alonso, J. Barbarán, J. Chen, M. Díaz, L. Llopis, B. Rubio, Middleware and communication technologies for structural health monitoring of critical infrastructures: a survey, Computer Standards & Interfaces 56 (2018) 83–100.

[2] M. Díaz, C. Martín, B. Rubio, State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing, Journal of Network and Computer Applications 67 (2016) 99–117.

[3] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, August 13–17, Helsinki, Finland, ACM, 2012, pp. 13–16.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.

[5] R. Morabito, V. Cozzolino, A.Y. Ding, N. Beijar, J. Ott, Consolidate iot edge computing with lightweight virtualization, IEEE Netw 32 (1) (2018) 102–111.

[6] D. von Leon, L. Miori, J. Sanin, N. El Ioini, S. Helmer, C. Pahl, R. Buyya, S. N. Srirama. Fog and Edge Computing: Principles and Paradigms, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2019, pp. 145–170.

[7] C. Dupont, R. Giaffreda, L. Capra, Edge computing in iot context: Horizontal and vertical linux container migration. 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June, IEEE, 2017, pp. 1–4.

[8] H. Cao, M. Wachowicz, C. Renso, E. Carlini, Analytics everywhere: generating insights from the internet of things, IEEE Access 7 (2019) 71749–71769, https://doi.org/10.1109/ACCESS.2019.2919514.

[9] J. Clemente, M. Valero, J. Mohammadpour, X. Li, W. Song, Fog computing middleware for distributed cooperative data analytics. IEEE Fog World Congress, FWC 2017, Santa Clara, CA, USA, October 30, - Nov. 1, 2017, IEEE, 2017, pp. 1–6.

[10] S. Tuli, N. Basumatary, R. Buyya, Edgelens: deep learning based object detection in integrated iot, fog and cloud computing environments, CoRR abs/1906.11056 (2019).

[11] D. Sinha, K. Doshi, M.R. Babu, An efficient approach to civil structures health monitoring using fog computing as clusters through 5g network environment, Adv Syst Sci Appl 18 (3) (2018) 123–143.

[12] J. Kharel, H.T. Reda, S.Y. Shin, Fog computing-based smart health monitoring system deploying lora wireless communication, IETE Technical Review 36 (1) (2019) 69–82.

[13] G. Ortiz, M. Zouai, O. Kazar, A.G. de Prado, J. Boubeta-Puig, Atmosphere: context and situational-aware collaborative iot architecture for edge-fog-cloud computing, Computer Standards & Interfaces (2021) 103550, https://doi.org/10.1016/j.csi.2021.103550.

[14] B. Omoniwa, R. Hussain, M.A. Javed, S.H. Bouk, S.A. Malik, Fog/edge computing-based iot (feciot): Architecture, applications, and research issues, IEEE Internet of Things Journal 6 (3) (2019) 4118–4149, https://doi.org/10.1109/JIOT.2018.2875544.

[15] B. Alturki, S. Reiff-Marganiec, C. Perera, S. De, Exploring the effectiveness of service decomposition in fog computing architecture for the internet of things, IEEE Trans. Sustainable Comput. (2019).

[16] A. Rullo, E. Serra, J. Lobo, Redundancy as a measure of fault-tolerance for the internet of things: A review. Policy-Based Autonomic Data Governance [extended papers from the Second International Workshop on Policy-based Autonomic Data Governance, PADG@ESORICS 2018, September 6, 2018, Barcelona, Spain]., 2018, pp. 202–226, https://doi.org/10.1007/978-3-030-17277-0_11.

[17] A. Burrello, A. Marchioni, D. Brunelli, L. Benini, Embedding principal component analysis for data reduction in structural health monitoring on low-cost iot gateways. Proceedings of the 16th ACM International Conference on Computing Frontiers, CF 2019, Alghero, Italy, April 30, - May 2, 2019., 2019, pp. 235–239, https://doi.org/10.1145/3310273.3322822.

[18] D. Todolḍ-Ferrandis, J. Silvestre-Blanes, S. Santonja-Climent, V. Sempere-Paya, J. Vera-PḚrez, Deploy&forget wireless sensor networks for itinerant applications, Computer Standards & Interfaces 56 (2018) 27–40, https://doi.org/10.1016/j.csi.2017.09.002.

[19] S. Tuli, N. Basumatary, R. Buyya, Edgelens: Deep learning based object detection in integrated iot, fog and cloud computing environments. 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Nov 21-22, Mathura, India, IEEE, 2019, pp. 496–502.

[20] A.I. Ozdagli, X. Koutsoukos, Machine learning based novelty detection using modal analysis, Comput.-Aided Civ. Infrastruct. Eng. 34 (12) (2019) 1119–1140, https://doi.org/10.1111/mice.12511.

[21] J. Haxhibeqiri, E. De Poorter, I. Moerman, J. Hoebeke, A survey of lorawan for Iot: From technology to application, Sensors 18 (11) (2018) 3995.

[22] A. Abdaoui, T.M. El Fouly, M.H. Ahmed, Impact of time synchronization error on the mode-shape identification and damage detection/localization in wsns for structural health monitoring, Journal of Network and Computer Applications 83 (2017) 181–189.

[23] R. Morabito, R. Petrolo, V. Loscri, N. Mitton, Legiot: a lightweight edge gateway for the internet of things, Future Generation Computer Systems 81 (2018) 1–15.

[24] E. KEMER, R. SAMLI, Performance comparison of scalable rest application programming interfaces in different platforms, Computer Standards & Interfaces 66 (2019) 103355, https://doi.org/10.1016/j.csi.2019.05.001.

[25] Z.-F. Fu, J. He, Modal analysis, Elsevier, 2001.

[26] E. Cañete, J. Chen, M. Diaz, L. Llopis, B. Rubio, Wireless sensor networks and structural health monitoring: experiences with slab track infrastructures, Int. J. Distrib. Sens. Netw. 15 (3) (2019), https://doi.org/10.1177/1550147719826002, 1550147719826002.

[27] R. Brincker, L. Zhang, P. Andersen, Modal identification of output-only systems using frequency domain decomposition, Smart Mater. Struct. 10 (3) (2001) 441.

[28] P.C. Chang, A. Flatau, S. Liu, Health monitoring of civil infrastructure, Structural health monitoring 2 (3) (2003) 257–267.

[29] Openmodal - a measurement, analysis and visualisation software for structural dynamics analysis, (Available online: http://www.openmodal.com/). (accessed on 27 October 2020).

[30] C. Martín, D.R. Torres, M. Díaz, B. Rubio, Fogpi: A portable fog infrastructure through raspberry pis. 9th Mediterranean Conference on Embedded Computing (MECO’2020), 8-11 June, Budva, Montenegro, IEEE, 2021.

[31] R. Brincker, L. Zhang, Frequency domain decomposition revisited. Proc. 3rd Int. Operational Modal Analysis Conf.(IOMAC09), 4–6 May, Portonovo, Italy, 2009, pp. 615–626.

[32] R. Gupta, D. Reebadiya, S. Tanwar, 6G-enabled edge intelligence for ultra -reliable low latency applications: vision and mission, Computer Standards & Interfaces 77 (2021) 103521, https://doi.org/10.1016/j.csi.2021.103521.

[33] C. Martín, D. Garrido, B. Rubio, M. Díaz, From the edge to the cloud: Enabling reliable iot applications. 7th International Conference on Future Internet of Things and Cloud (FiCloud 2019), 26–28 August, Istanbul, Turkey, IEEE, 2019, pp. 17–22.

[34] D. Corral-Plaza, I. Medina-Bulo, G. Ortiz, J. Boubeta-Puig, A stream processing architecture for heterogeneous data sources in the internet of things, Computer Standards & Interfaces 70 (2020) 103426, https://doi.org/10.1016/j.csi.2020.103426.

[35] H.-C. Chen, I. You, C.-E. Weng, C.-H. Cheng, Y.-F. Huang, A security gateway application for end-to-end m2m communications, Computer Standards & Interfaces 44 (2016) 85–93, https://doi.org/10.1016/j.csi.2015.09.001.

[36] Y. Liang, F. He, X. Zeng, 3D mesh simplification with feature preservation based on whale optimization algorithm and differential evolution, Integr. Comput. Aided Eng. 27 (4) (2020) 417–435, https://doi.org/10.3233/ICA-200641.

[37] Y. Wu, F. He, D. Zhang, X. Li, Service-oriented feature-based data exchange for cloud-based design and manufacturing, IEEE Trans. Serv. Comput. 11 (2) (2018) 341–353, https://doi.org/10.1109/TSC.2015.2501981.

[38] X. Zhao. New methods for structural health monitoring and damage localization, University of Sheffield, 2015. Ph.D. thesis.