sensors

MDPI

Article

BFT-PM10: A Resilient Distributed Architecture for Real-Time Air Quality Forecasting

Sergio Medina [1,2], Christian Fernández-Campusano [1,*], Humberto Verdejo-Fredes [1], Cristhian Becker [1] and Raúl Carrasco [3]

[1] Department of Electrical Engineering, Faculty of Engineering, University of Santiago de Chile (USACH), Santiago 9170124, Chile; sergio.medina.r@usach.cl (S.M.)
[2] Escuela de Ingeniería y Recursos Naturales, DUOC-UC, Santiago 8970117, Chile
[3] Departamento de Contabilidad y Gestión Financiera, Facultad de Administración y Economía, Universidad Tecnológica Metropolitana, Santiago 7500998, Chile; rcarrascoa@utem.cl
* Correspondence: christian.fernandez@usach.cl

Abstract

Air quality monitoring systems often suffer from frequent disruptions caused by unreliable sensor infrastructure, connectivity failures, and vulnerability to faulty or malicious nodes. To address these limitations, this study presents a conceptual fault-tolerant architectural framework for urban PM_{10} monitoring and short-term forecasting based on distributed Wireless Sensor Networks (WSNs). The proposed architecture integrates Edge Computing, Machine Learning (ML), and Byzantine Fault Tolerance (BFT) consensus within a three-tiered layout: IoT sensors, edge nodes, and cloud analytics. A lightweight, distributed leader-election mechanism is simulated to evaluate leadership convergence, availability, and workload distribution under dynamic network topologies. Simulation results across small-scale topologies (10–15 nodes) demonstrate stable leader-selection convergence within 5 s with balanced follower energy overhead under crash and omission events. Furthermore, multiple forecasting models (Linear Regression, Random Forest, Support Vector Regression, XGBoost, and ARIMAX) were retrospectively evaluated using historical daily PM_{10} lags and cumulative rainfall records from the O'Higgins Park monitoring station in the Chilean Metropolitan Region, with XGBoost achieving the lowest test error ($R^2 = 0.594$, RMSE = 17.681 $\mu g/m^3$), closely followed by Linear Regression ($R^2 = 0.586$, RMSE = 17.868 $\mu g/m^3$). By coupling BFT-oriented edge coordination with predictive analytics, this proof-of-concept design establishes a structured roadmap toward resilient environmental sensor networks.

Keywords: Byzantine fault-tolerance; distributed Wireless Sensor Network; edge-computing; machine learning prediction; PM_{10} monitoring

Check for updates

Academic Editors: Nyothiri Aung, Sahraoui Dhelim and Chaker Abdelaziz Kerrache

Received: 30 September 2026
Revised: 13 November 2026
Accepted: 18 November 2026
Published: 22 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

A distributed data fusion model integrates and analyses the information generated by multiple nodes or sensors located at different points within a network [1], thereby avoiding the need to rely on all measurements being concentrated on a single central server. In this approach, each node can perform local functions such as data acquisition, filtering, validation, pre-processing or inference [2,3], whilst data, extracted features or partial decisions are exchanged between neighbouring nodes or edge computing platforms to build a more complete, consistent and reliable representation of the monitored phenomenon [4]. The fusion of information from heterogeneous sources improves the accuracy of estimates, detects

anomalous readings, reduces the uncertainty associated with individual measurements and increases system availability in the event of sensor failures or connectivity interruptions [5]. Furthermore, by processing information close to its point of generation, this model helps to reduce response latency, bandwidth consumption and the exposure of sensitive data–characteristics that are particularly relevant for Internet of Things (IoT) applications involving environmental monitoring and real-time decision-making.

The aim of this model is to combine redundant and complementary measurements to achieve a more accurate and reliable estimate of the observed event. Each node processes a segment of the data independently and sends only the relevant results, decisions, or characteristics, enabling the system to make collective decisions [6]. This approach improves scalability and can increase fault tolerance in distributed sensor networks. In contexts such as PM10 forecasting, this type of model allows combining measurements from multiple geographically dispersed sensors to produce a more reliable prediction than one obtained from a single station or sensor [7,8].

The constant advancement of communication, automation and information systems, alongside the surge in the use of artificial intelligence techniques [9], enables the creation of a suite of associated systems [10], opening up a spectrum of possibilities for the use of Wireless Sensor Networks (WSN) [11]. Given this, and within the framework of the Smart City concept [12], the monitoring of variables that may affect the well-being of the entire community is of paramount importance.

In a trusted system for real-time PM10 monitoring, integrating a reliable Byzantine-fault-tolerant communication model with advanced data fusion techniques in distributed WSN substantially improves the accuracy, robustness, and resilience of PM10 concentration forecasting in the presence of malicious or faulty nodes.

This work aims to design and evaluate a reliable, Byzantine-fault-tolerant communication model, combined with data fusion techniques, to improve the accuracy and robustness of PM10 forecasting in distributed WSN.

Contributions:

- Develop a communication protocol capable of withstanding Byzantine faults, enabling the identification, isolation, and management of compromised nodes within the sensor network.
- A distributed data fusion model was proposed that efficiently combines various sensor measurements, taking into account uncertainty, noise and potentially harmful data.
- The performance of the proposed system was evaluated using metrics such as predictive accuracy, latency and resilience, and these findings were compared with conventional methods in both simulated and retrospective historical datasets from an urban air quality monitoring station.

1.1. Background

PM10 refers to inhalable particulate matter with an aerodynamic diameter of 10 micrometres or less. The importance of measuring this environmental parameter stems from its prolonged atmospheric persistence. This leads to deep penetration into the human respiratory tract and poses significant health risks, particularly for vulnerable populations [13].

1.1.1. From a Health Perspective, the Measurement of PM10 Particles

Measuring PM10 particles (particulate matter with a diameter of less than 10 micrometres) in heavily congested areas is a critical public health tool, as it enables the monitoring of a pollutant that has a direct impact on urban morbidity and mortality [14]. In areas with heavy traffic and industrial activity, levels of these particles soar due to the burning of fuels, wear and tear on brakes and tyres, and dust resuspended from the streets.

The spatial and temporal distribution of PM10 concentrations is governed by factors including vehicular emissions, combustion activities, dust resuspension, and local meteorological conditions. Consequently, monitoring systems must be capable of accurately capturing these variations and promptly detecting pollution episodes [15].

From a health perspective, exposure to PM10 is linked to heightened respiratory symptoms, the exacerbation of chronic diseases, and an elevated risk of hospitalisation. Consequently, controlling PM10 is a priority in air quality management. Therefore, the scientific literature highlights the need for robust monitoring and analysis methodologies to support prevention and mitigation decisions [16,17].

Figure 1 shows the locations of the existing monitoring stations in the National Air Quality Information System (SINCA) Metropolitan Region [18]. Although the location of these stations provides an overview of the city, it does not cover areas such as the northern districts (Conchali, Renca and Huechuraba) or the western districts (Maipu, Padre Hurtado), amongst other areas with large populations and industrial activity, leaving these areas without data. Furthermore, it must be borne in mind that not all areas have a similar distribution of population, industrial zones, traffic, green spaces and parks, etc., which means that pollution levels vary.

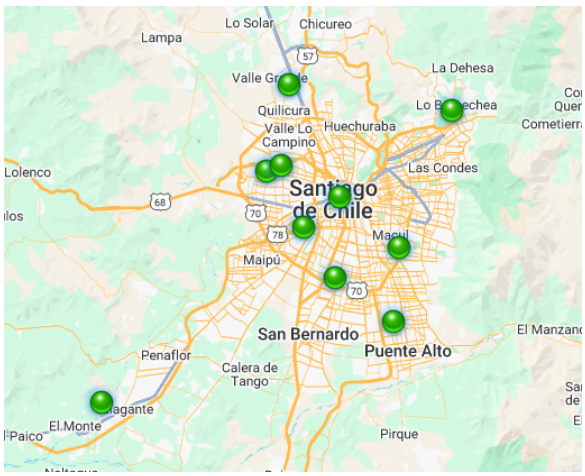


Figure 1. Locations of SINCA stations.

As stipulated by the regulations, three levels of danger are defined in Chile, depending on the amount of PM10, based on 24-h concentrations:

- Alert: 80 to 110 μg/m³N.
- Pre-emergency: 110 to 170 μg/m³N.
- Emergency: 170 μg/m³N or higher.

With the development of this information network, and the collection of data and historical records—which are in the public domain—it is possible to develop a set of coordinated tools that enable historical data analysis, as indicated in Salini Calderón and Pérez Jara [19], it is possible to develop artificial intelligence, using Machine Learning (ML) techniques, with which to predict the behaviour of environmental pollution levels in the metropolitan region, based on existing historical data. This would enable the implementation of preventive measures capable of reversing trends and, in this way, preventing an increase in respiratory diseases.

1.1.2. Distributed Systems

The concept of a distributed computer system was first introduced by Lamport [20], who presented the paradigm of a physically distributed system comprising distinct compo-

nents capable of processing information in part, exchanging data with other computing centres, and thereby jointly solving highly complex problems. Most distributed systems allow for the simultaneous execution of multiple processes; this simultaneous execution of processes within a system can be disrupted if some of them cease to function, whether due to a crash, a crash followed by recovery, or network disconnections. However, some of them may remain active and continue to function. These are the correct processes that remain operational and ensure connectivity to achieve synchronisation and consistency.

According to Laprie [21], distributed systems must possess dependability, which is defined as the ability of a distributed system to provide reliable services. This property can be represented by various attributes, including system availability, reliability, security, information protection and system sustainability, amongst others. Of the attributes mentioned, both availability and reliability are critical in distributed applications. Availability is described as a system's ability to maintain a service correctly, regardless of any faults that may occur.

1.1.3. Ubiquitous Systems

Ubiquitous systems represent an evolution of distributed systems. According to Weiser [22], ubiquitous systems are defined as those that allow computing to be integrated into daily life so naturally that people are unaware of their presence. Weiser [22], ubiquitous computing is about creating environments in which computers are as invisible and ever-present as people. Lyytinen and Yoo [23] expand on this definition by highlighting the importance of adaptability and the interconnection of devices within these systems. They state that "ubiquitous systems are characterised by their ability to provide contextually relevant services, dynamically adapting to the user's needs".

Satyanarayanan [24] highlights the ubiquity of technology in these systems, stating that 'ubiquitous computing involves the integration of devices and services into the environment in such a way that technology is accessible at all times and in all places'. Furthermore, the reliability of a system is its ability to ensure the continuous provision of its services. Thus, a distributed system is reliable if its behaviour is predictable despite partial failures and asynchronous time intervals. To achieve system reliability, a system must be robust, highly available, and guarantee that an application provides proper service despite failures, processing errors, or communication link failures; this makes a distributed system a fault-tolerant system.

2. Wireless Sensor Networks

WSNs have become a revolutionary technology that has transformed the way we interact with and monitor our physical environment. The versatility and adaptability of these networks have enabled their rapid proliferation, as innovative sensor technologies and advances in wireless communication have made it possible to deploy them in an ever-wider range of applications, from monitoring environmental conditions to managing industrial processes [25,26]. A WSN consists of devices capable of detecting the presence or changes in required environmental conditions (sensors), collecting and processing this information. This information is then encrypted and sent to a base station (Sink), using a wireless communication protocol (Zigbee, LoRaWAN, Wi-Fi, etc.) via a network topology.

WSNs are faced with the challenge of energy conservation, which makes efficient routing protocols crucial for prolonging network lifetime. In addition, delay time from sensors to the base station is critical in applications such as military, medical, and security monitoring systems [27,28].

2.1. Byzantine Fault-Tolerant Systems

The Byzantine Problem

The concept of Byzantine fault [29] derives from the Byzantine Generals' Problem, which is a problem in game theory. This problem arises when, hypothetically, two generals are besieging a castle, each commanding an army. Each army must decide whether to attack the castle or withdraw from the battle. To achieve victory, both armies must decide to attack at the same time or to withdraw. In summary, the problem is that the armies must reach a consensus to take the appropriate action.

This problem can be extended to N generals, which does not lead to the implementation of a system in which the generals can reach an agreement on a course of action. If we consider this problem in the context of distributed systems, with the generals acting as nodes in the system, it is possible to propose various algorithms that can meet this requirement.

There are a large number of faults that a node might experience at any given time. However, whether caused by external interference or simply integrity failures, these can be summarised into three types of failure for the Byzantine scenario: node crash, general omission, crash with recovery, and general failure; taken together, all of these are considered Byzantine faults (See Figure 2).

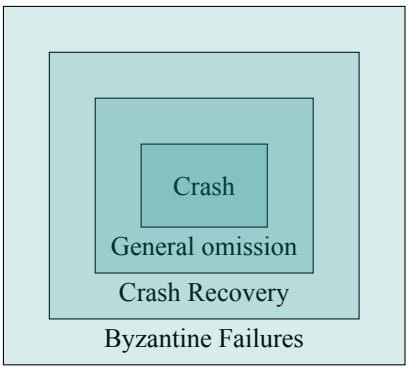


Figure 2. Classification of Byzantine faults in WSNs.

It can be noted that in the Byzantine fault model, faulty processes (also known as Byzantine processes) may deviate from their specification and execute any operation they wish, or even fail permanently [30]. Therefore, Byzantine processes may behave in a completely arbitrary manner. Thus, this model accounts for any type of behaviour exhibited by faulty processes, whether induced by third parties or by integrity failures [31].

In a WSN, a node may experience a large number of faults at any given time (see Figure 2). Whether caused by malicious attackers or simply by integrity failures, these, taken together, are considered Byzantine faults. On the other hand, fault tolerance in WSN is the network's ability to maintain its operational functions without interruption, even when some of its sensor nodes or communication links fail. Due to the deployment of WSNs in hostile environments and hardware limitations, this attribute is critical for ensuring the reliability of the data collected; hence the importance of Byzantine Fault Tolerance. A Byzantine process is a faulty process that can exhibit any behaviour. These processes may fail due to hardware or software issues, or to malicious attacks by third parties, which may also exhibit any behaviour, as illustrated in Figure 2.

2.2. Principle of Redundancy

As mentioned earlier, a distributed system must be fault-tolerant; such systems typically incorporate the principle of redundancy as a fundamental part of their design. Redundancy is a system's ability to replicate information across its critical components (whether

hardware, software, or both); in distributed systems, this redundancy means that each node shares information with all other nodes in the system, to ensure consistency and fault tolerance. Although redundancy is necessary, it is not sufficient on its own; according to Gartner [32], fault detection and correction are also required. One way to ensure correct replication of information across critical components is by addressing 'consensus problems' [33], which constitute a class of problems in distributed systems.

2.3. Distributed Consensus

This refers to the challenge of reaching an agreement amongst a group of nodes in a distributed system; this is a particularly significant challenge in fault-tolerant distributed systems, as discussed in Fischer [34]. This arises when reliable processes need to agree on a specific value based on the information available at each node. Once a consensus value has been reached, this problem becomes a leader selection problem; broadly speaking, this involves providing each node in the system with the necessary information, and ensuring that, when a response is required, it is the same node that sent the correct response. There are algorithms that enable these actions, such as the Paxos consensus algorithm [35], used in industry by various companies, including Google, IBM, VMware, etc.

2.4. Distributed Communication Models

The leader selection mechanism in a distributed system is to some extent dependent on timing assumptions; this is because each individual node is a subsystem with a different internal clock signal [36]. By making these timing assumptions, it is possible to specify the model for studying the timing and order of events in a distributed system. As indicated in Samaan and Jeiad [37], there are two well-defined models: the Synchronous Model and the Asynchronous Model (See Figure 3).

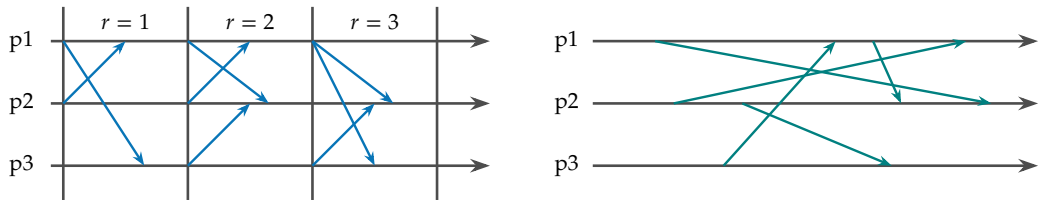


Figure 3. Synchronous execution (**left**) vs. asynchronous execution (**right**).

2.4.1. Synchronous Model

The synchronous model has explicit time bounds [38]; that is, a message is delivered within a bounded time, and the relative computation speed within a process is known. A synchronous distributed system is robustly based on message delivery and computation times. The main characteristics of the synchronous system are:

- The time required to execute each step of a process is known and limited. There are upper and lower time limits.
- Each message sent is received within a limited time.
- Each clock has a known and bounded deviation from real time.

The main problem with the synchronous system is that the bounds must be defined in such a way that they behave correctly under worst-case conditions. Otherwise, the assumptions underlying the system model may be violated, and the algorithm may consequently cease to be correct.

2.4.2. Asynchronous Model

In the asynchronous model [39,40], there are no upper bounds on message transmission delays or on the relative speeds of the processes; in other words, there are no timing assumptions. One difficulty implicit in this model is how to guarantee the system's safety and liveness properties. The impossibility of resolving the consensus problem deterministically when a process fails, known as the impossibility result [40], is based on the fact that it is sometimes impossible to distinguish between a process that has failed and a process that is taking a long time to respond to a message.

Some of the advantages of this model are:

- It is better suited to the rapidly growing and variable load on networks, in terms of processes and communication links.
- Algorithms that operate in this model always remain correct (a safety property) when run on any other system model.

2.4.3. Partially Synchronous Model

The partially synchronous model, presented by Dolev et al. [41], considers two types: in the first, time attributes are bounded, but these bounds are unknown. In the second, the time attributes are limited and known, but consensus is only reached after an unknown stabilisation interval; this is demonstrated by Dwork et al. [42], who shows that the consensus problem can be solved in both models, provided there is a majority of correct nodes in the system.

2.5. Fault Detectors

Chandra and Toueg [43], as a solution to the problem posed by [40] for asynchronous models, propose unreliable fault detectors; these are abstract models located at each node in the system, which allow the assumptions of synchronisation to be encapsulated within it. The function of these detectors is to provide (potentially incorrect) information regarding the operational state of the other nodes in the system. With this information provided, eight classes of detectors are defined, classified according to the information delivered, guaranteeing two properties: integrity and accuracy. With this, it is possible to solve the consensus problem using any unreliable fault detector.

Among the proposed detectors, one in particular—the Omega detector [43]—is one of the most interesting; it is the weakest detector capable of solving the distributed consensus problem, assuming a majority of correct nodes in the system. This detector asserts that there is a certain time; once this time has elapsed, all correct nodes believe in the same node. In this way, this detector provides a solution to the leader election problem [44].

In the strict sense of a communication system that is 100% stable and reliable, a healthy node should never fail. Once a failure occurs, that node should be marked as faulty and communication with it should not be re-established. However, in the fault model of [44], a node is considered faulty when it experiences a crash and does not recover, or when it fails frequently. Fernández-Campusano et al. ([45,46]) establishes a leader election system that classifies as correct nodes those that fail and recover, and those that never fail.

2.6. Leader Selection Algorithm

To design an algorithm that addresses the leader election problem, one must determine the number of nodes that may fail for the various possible Byzantine faults; bearing this in mind, it must be assumed that the number of failed nodes is a small percentage, both for crash-recovery and for message transmission and reception failures (see Figure 2). In Table 1, taken from [47], the formula $3f + 1 = n$ is established for a Byzantine system (among many others), where n is the total number of entities communicating and f is

the maximum number of faulty nodes that the system can tolerate before ceasing to be Byzantine. Based on this, it is calculated that for desired systems with $n = 7$, there can be a maximum of $f = 2$ faulty nodes.

Table 1. Byzantine protocols tolerant of f faulty nodes.

Name	Network Model	Nodes	Type
B-Chain-3	Partial synchronisation	$3f + 1$	Chain
Zyzzva	Partial synchronisation	$3f + 1$	Broadcast
PBFT	Weak synchronisation	$3f + 1$	Broadcast
Q/U	Asynchronous	$5f + 1$	Quorum
HQ	Asynchronous	$3f + 1$	Quorum
Aliph-Chain	Asynchronous	$3f + 1$	Quorum/Chain/broadcast
Ben-Or et al.	Synchronous	$4f + 1$	Random/Quorum
Mostéfaoui et al.	Asynchronous	$3f + 1$	Random/Broadcast
AER	Asynchronous	$3f + 1$	Random/Quorum
Patra et al.	Asynchronous	$3f + 1$	Random/Broadcast
HoneBaderBFT	Asynchronous	$3f + 1$	Random/Broadcast

2.7. Theoretical Foundations and Dimensions of the Byzantine Consensus

The design of Byzantine Fault Tolerance (BFT) protocols under the partial synchrony model requires strict limits to be set on the proportion of malicious or faulty nodes that the system can tolerate. In this network model, originally formulated by Dwork et al. [42], message delivery delays are bounded but unknown a priori until a Global Stabilisation Time (GST) is reached.

2.7.1. Lower Bound and the Quorum Intersection Theorem

To guarantee the fundamental properties of Safety (consistency amongst correct nodes) and Liveness (system progress), the total number of nodes n, in the presence of f nodes with Byzantine faults, must satisfy the necessary and sufficient condition set out in Equation (1):

$$n \geq 3f + 1 \quad \Longleftrightarrow \quad f \leq \lfloor \frac{n-1}{3} \rfloor \tag{1}$$

The formal proof of this bound follows from the operational requirements for quorums in networks where it is not possible to distinguish between a malicious node that omits messages and a legitimate node that is severely delayed by the network:

1. Liveness Guarantee (Availability): In the worst-case execution scenario, up to f nodes may stop responding or sending messages. Therefore, to prevent the protocol from becoming indefinitely blocked, a decision quorum Q must not require more than $n - f$ responses:

$$|Q| = n - f \tag{2}$$

2. Safety Guarantee (Consistency): To prevent contradictory decisions or forks (equivocation), any pair of quorums Q_1 and Q_2 must intersect at at least $f + 1$ nodes. This ensures that the intersection contains at least one correct node that preserves the state:

$$|Q_1 \cap Q_2| \geq f + 1 \tag{3}$$

Applying the inclusion-exclusion principle for sets to Equation (3):

$$\begin{aligned} |Q_1 \cap Q_2| &= |Q_1| + |Q_2| - |Q_1 \cup Q_2| \geq (n - f) + (n - f) - n \\ &= n - 2f \end{aligned} \tag{4}$$

By substituting the lower limit of intersection, we obtain the fundamental inequality:

$$n - 2f \geq f + 1 \implies n \geq 3f + 1 \quad (5)$$

2.7.2. Seminal Literature and Historical Contributions

The strict bound $n \geq 3f+1$ has been validated and formalised by the leading researchers in distributed systems theory (See Table 2):

- Lamport et al. [29]: They first introduced the Byzantine Generals' Problem and proved that no algorithm exists for $n \leq 3f$ in communication channels that are not authenticated by means of spoken messages.
- Dwork et al. [42]: They laid the foundations for the partial synchrony model and demonstrated the lower bounds on complexity and fault tolerance in networks with uncertain timing.
- Castro and Liskov [48]: They demonstrated the practical feasibility of this bound by developing the PBFT (Practical Byzantine Fault Tolerance) protocol, showing that a quorum of $2f + 1$ out of $3f + 1$ nodes enables efficient distributed agreement with low latency.

Table 2. Quorum parameters for Byzantine fault tolerance under partial synchronisation.

System Parameter	Formal Expression	Case Study ($f = 1$)
Minimum total number of nodes (n)	$3f + 1$	4 nodes
Maximum number of Byzantine nodes (f)	$\lfloor \frac{n-1}{3} \rfloor$	1 node
Minimum quorum size (Q)	$2f + 1$	3 nodes
Minimum quorum intersection	$f + 1$	2 nodes

2.7.3. Byzantine Errors and Other Errors

In distributed systems theory, the hierarchy of fault models states that the Byzantine (arbitrary) model is a superset that encompasses all restricted or benign fault models. Consequently, a Byzantine node can deliberately mimic or disguise itself as any benign fault in order to go undetected by failure detectors:

- General Omission Simulation: The Byzantine node selectively discards or delays certain messages destined for a subset of peers (omission-based equivocation). To the rest of the network, this behaviour is indistinguishable from buffer congestion or packet loss on the communication link.
- Crash-Recovery Simulation: The node suspends its participation at strategic moments (for example, during the commit phase) and subsequently reappears requesting state catch-up, citing a temporary loss of volatile memory (RAM) or a hardware reset. Crash Simulation.
- Definitive (Crash/Fail-Stop): The node abruptly ceases all message transmission (fail-silent) precisely when its vote is required to complete the quorum ($2f + 1$), simulating a catastrophic hardware failure to force a degradation in liveness or trigger a view-change.

Equivalence and inclusion relations between fault models (demonstrating how algorithms designed for arbitrary faults, by definition, cover cases of omission and crash) were rigorously formalised by Hadzilacos and Toueg [38].

3. Machine Learning Techniques

3.1. Linear Regression Model

The multiple linear regression model [49] was chosen because of its ability to provide an interpretative framework that quantifies the contribution of the pollutants under consideration—such as PM10 with a time lag and precipitation—as well as meteorological variables to the concentration of particulate matter. This model [50] not only allows us to estimate how PM concentrations vary under different meteorological conditions, but is also crucial for assessing the impact of climate change on these concentrations. Furthermore, multiple linear regression is used as a benchmark [51] due to its simplicity and ease of interpretation, although it is acknowledged that it may not capture non-linear relationships between visibility and concentrations. Nevertheless, it provides a valuable point of comparison for other models.

The estimation of $PM10_t$ is shown in the following equation:

$$\begin{aligned} PM10_t = {} & \beta_0 + \beta_1 PM10_{t-1} + \beta_2 PM10_{t-2} \\ & + \beta_3 PM10_{t-3} + \beta_4 PM10_{t-7} \\ & + \beta_5 PM10_{t-14} + \beta_6 24 - \text{hourRainfall}_t + \varepsilon_t \end{aligned} \tag{6}$$

where:

$$\varepsilon_t \sim N(0, \sigma^2).$$

Each coefficient β_j represents the expected change in $PM10_t$ in response to a unit increase in the corresponding predictor, holding all other variables constant.

In particular, if:

$$\beta_6 < 0,$$

then higher cumulative rainfall over 24 h is associated with an expected decrease in $PM10_t$.

3.2. Random Forest Model

The Random Forest (RF) technique, which has been studied and applied in various fields, such as environmental modelling and weather forecasting [52], is a supervised ML algorithm designed for both classification and regression. Its robustness lies in its ability to combine multiple decision trees, each trained on a bootstrap random sample of the original dataset [53]. This methodology streamlines the learning process by minimising the correlation between individual models and increasing their robustness to multicollinearity, thereby promoting superior predictive performance in complex contexts [54]. RF regression has found various successful applications in forest fire prediction and air quality studies, capitalising on its ability to capture non-linear relationships and interact effectively with multiple environmental factors [55].

Random Forest is an ensemble of regression trees trained on bootstrap samples. The final prediction is the average of the predictions from all the trees:

$$\widehat{PM10}_t = \frac{1}{B} \sum_{b=1}^{B} T_b(\mathbf{x}_t),$$

where B is the number of trees and $T_b(\mathbf{x}_t)$ is the prediction output by the b-th tree.

If 500 trees are used, the expression becomes:

$$\widehat{PM10}_t = \frac{1}{500} \sum_{b=1}^{500} T_b(\mathbf{x}_t).$$

Each tree splits the space of predictors using rules of the following type:

$$\begin{matrix} PM10_{t-1} \leq c_1, \\ 24 - hourRainfall_t \leq c_2, \\ PM10_{t-14} \leq c_3. \end{matrix}$$

If $\mathbf{x}_t$ belongs to a terminal region R_j, the tree's prediction is the average of the observed values in that region:

$$T_b(\mathbf{x}_t) = \bar{Y}_{R_j} \quad \text{si} \quad \mathbf{x}_t \in R_j.$$

Therefore, Random Forest predicts $PM10_t$ as the average of many decision rules generated by different trees.

3.3. Extreme Gradient Boosting Model

The XGBoost regression algorithm, an extreme gradient booster (XGBoost is notable for its advanced regularisation and its capacity for distributed computing, as noted in [56]), implements various techniques that optimise the efficient and accurate prediction of continuous or discrete variables, characteristics of the ML approach known as gradient boosting [57]. This method, based on decision trees that are constructed sequentially and adjust the model along the gradient direction calculated from past residual errors, has been widely used for classification and regression problems due to its ability to handle sparse data and process it efficiently [58].

Furthermore, the method is characterised by its robustness through the inclusion of a measure of model complexity in the objective function, which encompasses both the traditional loss function and the additional complexity associated with the model [58]. This approach contrasts with the gradual and systematic evolution of successive weak classifiers based on decision trees, where each new component attempts to correct the errors of its predecessors [57,59], thus reflecting a clear evolution of the gradient booster towards a more effective and efficient model.

$$\widehat{PM10}_t = \sum_{m=1}^{M} f_m(\mathbf{x}_t),$$

where M denotes the total number of trees and f_m denotes the mth tree. The model is estimated by minimising a regularised objective function:

$$\mathcal{L} = \sum_{t=1}^{n} l\left(PM10_t, \widehat{PM10}_t\right) + \sum_{m=1}^{M} \Omega(f_m).$$

In the case of least squares regression, the loss function can be expressed as follows:

$$l\left(PM10_t, \widehat{PM10}_t\right) = \left(PM10_t - \widehat{PM10}_t\right)^2.$$

The tree complexity penalty is expressed as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2,$$

where T represents the number of leaves in the tree, w_j denotes the weight of leaf j, γ penalises the structural complexity of the tree and λ penalises high weights. In summary, XGBoost estimates a non-linear function:

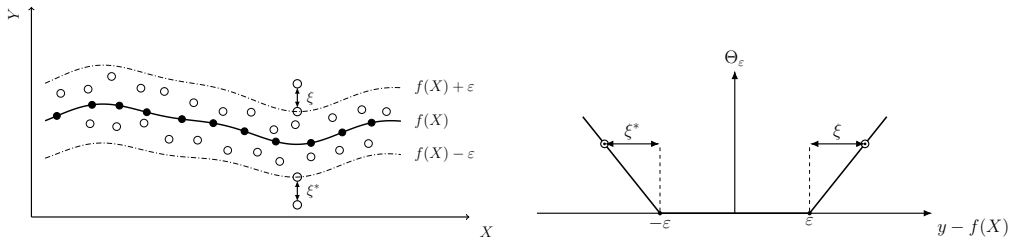
$$\widehat{PM10}_t = F(\mathbf{x}_t),$$

where F is a function learnt using boosting.

3.4. Support Vector Regression

Support Vector Regression (SVR) emerged as an extension of the support vector machine, although it is specifically designed for regression tasks [60]. Rather than classifying data into classes, SVR is used to identify an optimal hyperplane that accurately predicts continuous values by utilising different types of kernels [61], including linear, polynomial and Gaussian function of a radial base (RBF) kernels. The effectiveness of SVR is particularly evident in the prediction of environmental variables such as air quality, where its ability to handle complex non-linear relationships has led to its growing adoption in environmental epidemiology studies [62].

Figure 4 presents a conceptual diagram of the SVR used to predict $PM10_t$. It highlights the tolerance margin ε, the regression function and the observations that act as support vectors [63].



(a) (b)

Figure 4. Schematic diagram; (**a**) SVR, and (**b**) SVR using an ε-insensitive function.

The SVR can be represented as:

$$\widehat{PM10}_t = f(\mathbf{x}_t) = \mathbf{w}^\top \phi(\mathbf{x}_t) + b,$$

where $\phi(\mathbf{x}_t)$ maps the predictors into a higher-dimensional space, $\mathbf{w}$ represents the weight vector and b is the bias term. In the case of a radial kernel, the Gaussian kernel is used:

$$K(\mathbf{x}_i, \mathbf{x}_t) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_t\|^2\right).$$

The prediction of the radial SVR is expressed as:

$$\widehat{PM10}_t = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x}_t) + b,$$

where α_i and α_i^* act as Lagrange multipliers. Only certain observations have non-zero values for $\alpha_i - \alpha_i^*$; these observations are the support vectors. The SVR optimisation problem is formulated as:

$$\min_{\mathbf{w}, b, \xi_i, \xi_i^*} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*),$$

subject to:

$$\begin{aligned} PM10_i - f(\mathbf{x}_i) &\leq \varepsilon + \xi_i, \\ f(\mathbf{x}_i) - PM10_i &\leq \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* &\geq 0. \end{aligned}$$

Furthermore, the appropriate tuning of hyperparameters such as γ, ε and C is crucial to avoid overfitting and ensure good predictive performance [64]. All of this highlights the importance of statistical learning in the accurate prediction of environmental variables, as demonstrated by studies that combine theory with practical applications [65].

3.5. ARIMAX Model

The ARIMAX model, which incorporates exogenous variables that are crucial for forecasting time series dynamics, is establishing itself as an innovative development in the field of forecasting. Its design is inspired by the fundamentals of the standardised ARIMA model [66], and its core innovation lies in the incorporation of external factors that enhance the model's ability to capture trends and seasonal variations present in the data, thereby achieving more accurate forecasts. Furthermore, this methodology stands out not only for its predictive accuracy but also for its robust analytical contributions, a fundamental aspect when integrating autoregressive and differencing components alongside exogenous variables [67]. Furthermore, various studies have demonstrated the effectiveness of ARIMAX in forecasting atmospheric pollutant concentrations, highlighting its practical applicability and its ability to improve existing models that include exogenous variables [68]. Taken together, these methodologies offer a comprehensive approach that addresses dynamic aspects of time series with increasingly recognised accuracy and robustness.

The ARIMAX model combines a regression with external explanatory variables and an ARIMA structure in the errors. For the case analysed, it can be represented in Equation (6).

The error term u_t follows an ARMA structure:

$$\phi(B)u_t = \theta(B)\varepsilon_t,$$

where B is the lag operator, such that:

$$BY_t = Y_{t-1}.$$

In general:

$$\left(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\right)u_t = \left(1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q\right)\varepsilon_t.$$

Equivalently:

$$u_t = \phi_1 u_{t-1} + \phi_2 u_{t-2} + \cdots + \phi_p u_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}.$$

Therefore, the ARIMAX model forecasts $PM10_t$ using lags of PM10, precipitation and an additional time correction for the errors.

3.6. Performance Metrics

In PM10 forecasting, performance metrics such as the Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and the Coefficient of Determination (R^2) are essential for assessing, in general terms, the bias, accuracy and performance of the proposed model [69–72].

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{7}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{9}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{10}$$

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{11}$$

4. The Proposal

In order to carry out this research effectively, we propose, first of all, to conduct a series of simulations, both for the proposed architecture (Figure 5) and to predict the concentration of particulate matter in the air using ML tools.

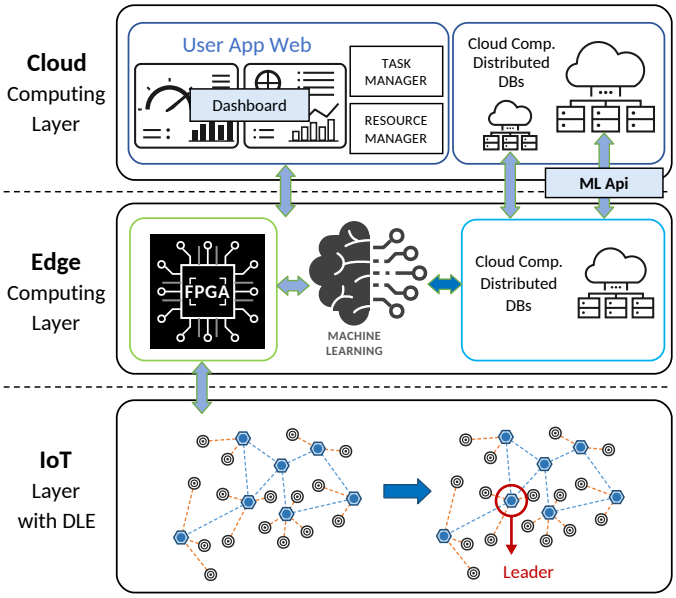


Figure 5. Proposed architecture.

4.1. Proposed Architecture

The architecture proposed for this is the one described in Vera et al. [73], which is divided into three distinct layers (see Figure 5):

First, the cloud computing layer. Second, the edge computing layer. Third, the IoT network device layer, as shown in Figure 5. Each layer is described below:

- Cloud Computing Layer: This layer enables various user applications, such as control panels and indicators. The processed information and insights presented in this layer are derived from data in the IoT (WSN) layer, where the data is analysed using ML techniques to make predictions based on the measured PM10 levels, as well as meteorological variables that may alter environmental and ventilation conditions, thereby affecting particulate matter levels.
- Edge Computing Layer: In this layer, Field-Programmable Gate Array (FPGA) devices are considered as edge computing elements. These receive information from the lower layer (the IoT WSN network). They then process information from both the upper and lower layers and facilitate communication between the two layers. The FPGA devices (in the Edge Computing layer) process the data sent by the single correct leader from the lower layer.
- IoT Layer: This layer represents IoT devices; it is a network of WSN devices which, through the use of a Distributed Leader Election (DLE) algorithm, allows a single correct leader (or none) to be selected to send or receive the information that must be processed by the FPGA devices in the Edge Computing layer.

4.2. Communication Algorithm

At this stage, the algorithm proposed by [46] is adapted for Byzantine fault tolerance (see Table 1). All messages exchanged by the nodes are transmitted through reliable and authenticated communication channels [46]. Each message is encrypted before transmission

and decrypted upon reception; its origin and integrity are verified before the message is processed by the DLE algorithm. Therefore, messages that fail the corresponding cryptographic verification are discarded and are not used in the leader-election procedure. The following tasks are defined:

- Initialisation: As the first step of the algorithm, each process p begins execution of the algorithm without a designated leader. Concurrently, p sets up an individual timer for each process q in the system. As illustrated in Algorithm 1, this initialisation protocol is executed by every process p in two scenarios: (i) during its first start-up, and (ii) after each recovery following a failure. The purpose of this procedure is to prepare the set of variables that will allow process p to monitor both its own state and that of the other processes. It is important to note that this mechanism does not use persistent storage to record the number of post-failure recovery events.
- Task A: This task periodically executes the sequence $idMsgSent_p$, where p sends a message to check whether the receiver q is in the ALIVE state; a counter is incremented for each message sent. This task is used as a status indicator for the nodes present in the network.
- Task B: As soon as p receives the ALIVE message from node q, the penalty value that p holds in relation to the other nodes is updated, provided that the values received are correct. In this case, $idMsgReceived_p[q]$ is set to the correct value so that the message can be processed.
- Task C: When process ℓ_p receives a PENDING message from q, if the current penalty value of ℓ_p matches the value received in the message, then ℓ_p resends all 'pending' ALIVE messages between ℓ_p and q, to re-establish the connection between ℓ_p and q.
- Task D: If the timeout expires at node q and the ALIVE message is not received within the given time, the timeout is reset and the node's connection is refreshed. The task ends with a Leader update.
- UpdateLeader: In the UpdateLeader() procedure (Algorithm 2), p chooses a leader only if it is well-connected both for receiving and sending with a Byzantine quorum ($>2n/3$ processes), selecting from among the processes with which p is well-connected the process q with the lowest $penalty_p[q]$ and the highest energy value (using the process ID to break ties).

The BFT algorithm incorporated into the proposal is a component that was previously developed, tested and validated [46,74].

Therefore, the main contribution of this article does not lie in the formulation of a new BFT consensus mechanism, but rather in the definition of the functional architecture and proposed design strategy through which this algorithm will be integrated into the sensor nodes of the proposed system.

The significance of BFT lies in its ability to maintain the consistency, availability and integrity of a distributed system even when a fraction of its nodes exhibit arbitrary, erroneous or potentially malicious behaviour [75]. This property is particularly important in IoT scenarios, edge computing and distributed sensor networks, where devices may operate in heterogeneous environments, with variable connectivity, limited resources and exposure to physical failures, attacks or data tampering [76]. In this context, the BFT mechanism ensures that the system's collective decisions do not depend on the individual reliability of a single node, but rather on a process of validation and consensus amongst multiple participants.

Algorithm 1: Main Tasks

Task *Initialization*;
$recovered_p[p] \leftarrow 0$;
foreach $q \in \Pi \setminus \{p\}$ **do**
  $recovered_p[q] \leftarrow 0$;
  send$(RECOVERED, p)$ to q;
  $idMsgSent_p[q] \leftarrow 0$;
  $idMsgReceived_p[q] \leftarrow 0$;
  $wellConnectedIn_p[q] \leftarrow FALSE$;
  $wellConnectedOut_p[q] \leftarrow FALSE$;
  $leaderCandidates_p[q] \leftarrow FALSE$;
$\ell_p \leftarrow \perp$;
foreach $q \in \Pi \setminus \{p\}$ **do**
  $Timeout_p[q] \leftarrow 0$;
  reset $timer_p[q]$ to $Timeout_p[q]$;
start tasks: Task A, Task B, Task C, Task D and Task E;

Task Task A: When receive $(RECOVERED, q)$;
$recovered_p[q] \leftarrow recovered_p[q] + 1$;
$UpdateLeader()$;

Task Task B: Repeat every η time units;
if $\ell_p = \perp \ \vee \ \ell_p = p$ **then**
  foreach $q \in \Pi \setminus \{p\}$ **do**
    $idMsgSent_p[q] \leftarrow idMsgSent_p[q] + 1$;
    send$(ALIVE, p, recovered_p, idMsgSent_p[q], wellConnectedIn_p[q], \ell_p)$ to q;
else
  $idMsgSent_p[q] \leftarrow idMsgSent_p[\ell_p] + 1$;
  send$(ALIVE, p, recovered_p, idMsgSent_p[q], wellConnectedIn_p[q], \ell_p)$ to ℓ_p;

Task Task C: When receive $(ALIVE, q, recovered_q, idMsgSent_q, wellConnected_q, \ell_q)$;
if $idMsgSent_p[q] = idMsgSent_p[\ell_p] + 1$ **then**
  reset $timer_p[q]$ to $Timeout_p[q]$;
  foreach $q \in \Pi \setminus \{p\}$ **do**
    $recovered_p[q] \leftarrow \max(recovered_p[q], recovered_p[q])$;
  $idMsgReceived_p[q] \leftarrow idMsgSent_p$;
  $wellConnectedIn_p[q] \leftarrow TRUE$;
  $wellConnectedOut_p[q] \leftarrow wellConnected_q$;
  $leaderCandidates_p[q] \leftarrow (q = \ell_q)$;
  $UpdateLeader()$;
else
  send$(PENDING, p, idMsgReceived_p[q])$ to q;

Task Task D: When receive $(PENDING, q, idMsgReceived_q)$;
if $idMsgReceived_q < idMsgSent_p[q]$ **then**
  for $msgid \leftarrow idMsgReceived_q + 1$ **to** $idMsgSent_p[q]$ **do**
    send$(ALIVE, p, recovered_p, msgid, wellConnectedIn_p[q], \ell_p)$ to q;

Task Task E: Upon expiration of $Timer_p[q]$;
$Timeout_p[q] \leftarrow Timeout_p[q] + 1$;
$wellConnectedIn_p[q] \leftarrow FALSE$;
$leaderCandidates_p[q] \leftarrow FALSE$;
$UpdateLeader()$;

Algorithm 2: Leader Update

procedure *Update Leader*
- $C_p \leftarrow \{p\}$;
- **foreach** $q \in \Pi \setminus \{p\}$ **do**
  - **if** $wellConnectedIn_p[q] = \text{true} \ \wedge \ wellConnectedOut_p[q] = \text{true}$ **then**
    - Insert q into C_p;
- $\ell_p \leftarrow \perp$;
- $q \leftarrow$ select process $\in C_p$ with smallest $recovered_p[q]$, breaking ties using process identifiers;
- **if** $q = p$ **then**
  - **if** $|C_p| > 2n/3$ **then**
    - $\ell_p \leftarrow p$;
- **else**
  - **if** $leaderCandidates_p[q] = \text{true}$ **then**
    - $\ell_p \leftarrow q$;

4.3. Block Construction Diagram

Figure 6 shows the proposed functional block diagram for the development and proposed design of an FPGA-based edge computing platform, designed to integrate ML techniques, BFT mechanisms, sensors and communication modules. The architecture follows a design sequence that begins with the development and validation of the algorithmic components, continues with their adaptation to a reconfigurable hardware platform, and culminates in experimental evaluation using a proof-of-concept prototype.

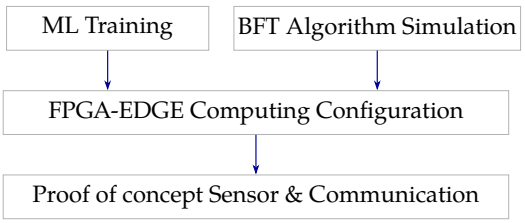


Figure 6. Block diagram of the proposed platform.

Firstly, the ML Training block corresponds to the training and validation phase of the machine learning model. This stage involves data acquisition, cleaning, selection and preparation, as well as the tuning of hyperparameters and the evaluation of the model's performance using metrics appropriate to the task at hand. The result is a trained and optimised model whose parameters, structure and inference requirements can guide a subsequent deployment on an FPGA platform.

The BFT Algorithm Simulation block represents the design and simulation phase of the Byzantine fault-tolerant algorithm. Its purpose is to analyse the behaviour of the consensus mechanism under adverse conditions, such as message loss, communication delays, node failures or the presence of participants acting maliciously. The simulation enables the determination of relevant operational parameters, such as the number of replicas, voting thresholds, consensus latency, the volume of messages exchanged, and the system's ability to maintain the integrity of distributed decisions. The results obtained at this stage define the functional configuration to be incorporated into the edge solution.

The outputs from the ML training and BFT simulation blocks converge at the central FPGA-Edge Computing Configuration block. This block forms the integration layer between the algorithmic design and the physical implementation. At this stage, the machine learning

model is adapted for efficient execution on the FPGA, taking into account constraints relating to logical resources, available memory, numerical precision, power consumption and inference latency. In addition, components associated with the BFT algorithm are integrated, such as message handling, state validation, voting mechanisms and the communication interfaces required for coordination between nodes. The FPGA enables low-latency local processing and reduces reliance on remote services, which is particularly relevant in IoT scenarios with response time or privacy requirements.

The 'Proof of Concept Sensors and Communication' block corresponds to the experimental implementation of the system. This layer incorporates the sensors responsible for capturing the physical or contextual variables of interest, together with the communication modules used to exchange data, decisions or consensus messages between the various devices. The acquired data is sent to the FPGA platform to execute the ML model inference locally and, where appropriate, to apply BFT mechanisms to ensure the consistency and reliability of distributed decisions. This stage enables a comprehensive assessment of the proposal's viability in a representative environment, taking into account metrics such as end-to-end latency, inference accuracy, availability, energy consumption, transmission rate, communication reliability and fault tolerance.

Consequently, the diagram sets out a four-stage, phased proposed design methodology: (i) training the machine learning model; (ii) simulating and validating the BFT algorithm; (iii) configuring and integrating the system into the edge computing FPGA platform; and (iv) validation using sensors and communications in a proof-of-concept prototype. This structure allows the activities of design, algorithmic validation, hardware implementation and experimental evaluation to be separated, facilitating system traceability and the reproducibility of results.

5. Results

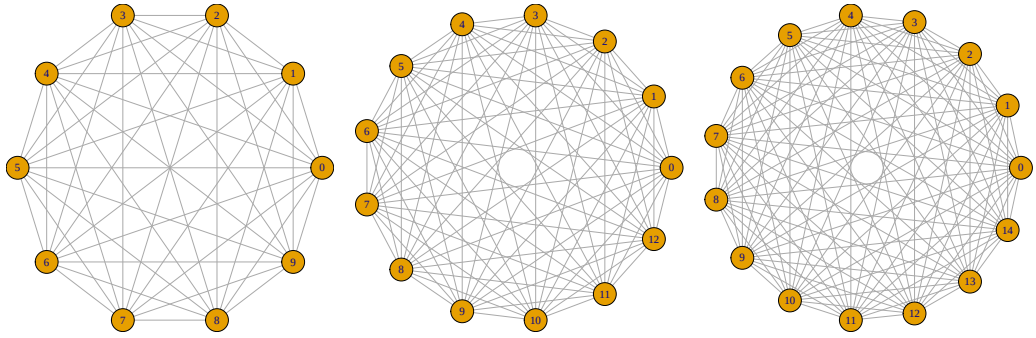
The configuration of the IoT network is defined by taking into account the fundamental mathematical condition of BFT systems, which states that, in order to tolerate up to f nodes exhibiting arbitrary, erroneous or malicious behaviour, the total number of nodes must satisfy the relationship $N \geq 3f + 1$, where N corresponds to the total number of participating nodes. Under this criterion, a network of seven nodes constitutes the minimum configuration capable of tolerating up to two Byzantine nodes; however, an architecture comprising ten sensor nodes is proposed, as this number allows for the tolerance of up to three Byzantine nodes whilst simultaneously increasing the spatial redundancy of measurements, the availability of information and the robustness of communications. The choice of ten nodes should not be understood solely as an extension of fault tolerance, but also as a functional decision to improve the reliability of environmental monitoring. By having multiple data sources, the system can cross-check readings, identify inconsistent values and mitigate the impact of faulty sensors, compromised nodes or connectivity interruptions. The relationship $N \geq 3f + 1$ allows sufficient quorums of correct nodes to be formed and prevents contradictory decisions.

5.1. Simulation of Transmission Algorithms

Based on Fernández-Campusano [46], Vera et al. [73], a series of simulations were carried out using OMNET++ version 6.0.3 on a computer equipped with an AMD Ryzen 8465HS processor and 16 GB of RAM.

These simulations involve using the proposed communication algorithms, in which the nodes transmit information using simple encryption, as shown in Figure 7a–c, in the form of a broadcast, whilst evaluating the penalty and energy consumption of each node in order to select the leader. These simulations cover three different scenarios, with 10, 13

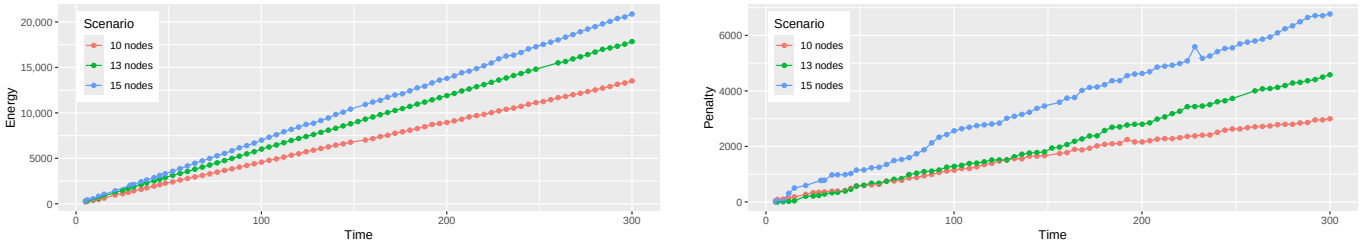
and 15 nodes respectively. In each simulation, the behaviour of the leader selection process, the messages sent by each node, the energy consumption and the penalty incurred by each node were monitored.



(a) Scenario 1: 10 Nodes (b) Scenario 2: 13 Nodes (c) Scenario 3: 15 Nodes

Figure 7. DLE algorithm test scenarios.

Analysis of the leader selection behaviour across all simulations shows that the system takes a maximum of 5 s to select the leader; although the leading node may fluctuate, the system remains stable and leaders are selected correctly. In Figure 8a,b, we illustrate the behaviour of the leaders' energy and penalty over time, maintaining stable behaviour over time.



(a) Energy consumption per node (b) Penalty for failure to receive or omission of messages

Figure 8. Leading performance on stage.

If we look at the graph comparing the energy consumption of the nodes in each simulation, we can see that the leaders consume more energy than the other nodes (Figure 9), since, as can be seen in the message transmission graphs, all nodes sent a similar number of messages, but energy consumption is lower in the non-leader nodes, with the exception of those nodes that acted as leaders for brief periods of time.

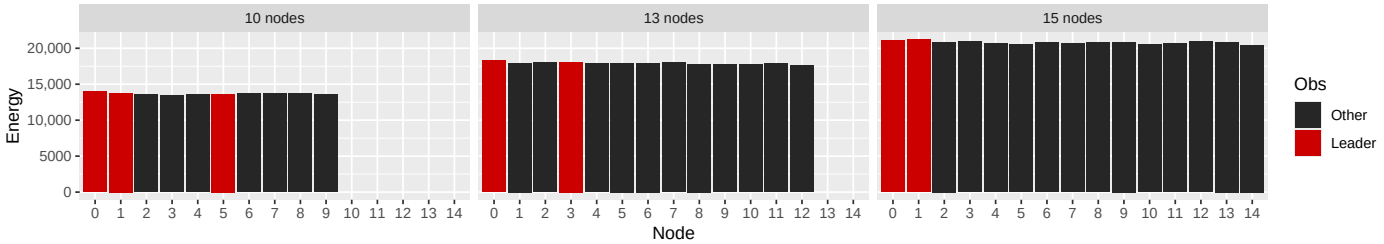


Figure 9. Energy consumption per node.

Similarly, as shown in Figure 10, we can see that the nodes that were leaders for the longest duration are those with the lowest penalty in all three proposed scenarios. Table 3 illustrates how the leader selection behaved in each scenario.

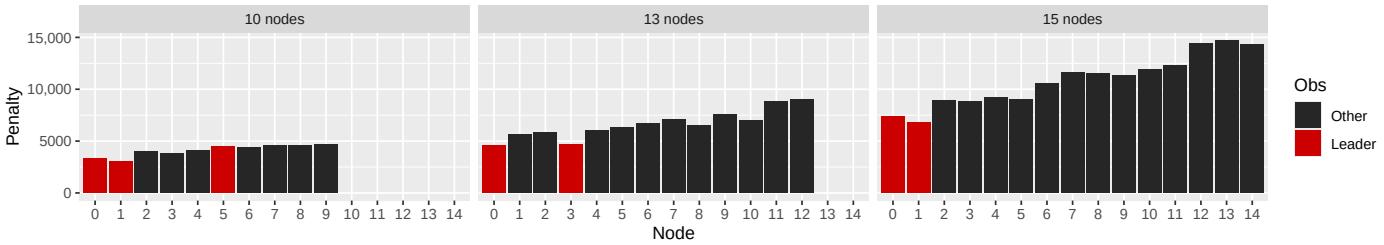


Figure 10. Penalty for failure to receive or omission of messages per node.

Table 3. Leadership behaviour over time in three scenarios.

Time [s]	Leader 10 Node	Leader 13 Node	Leader 15 Node
1	none	none	none
2	none	none	none
4	none	none	Node 5
5	Node 5	Node 0	Node 5
10	Node 5	Node 0	Node 5
12	Node 5	Node 0	Node 5
16	Node 0	Node 0	Node 5
20	Node 0	Node 0	Node 0
24	Node 0	Node 0	Node 0
⋮	⋮	⋮	⋮
154	Node 0	Node 0	Node 0
155	Node 0	Node 0	Node 0
156	Node 1	Node 0	Node 0
157	Node 1	Node 0	Node 1
158	Node 1	Node 0	Node 1
162	Node 1	Node 0	Node 1
166	Node 1	Node 0	Node 1
⋮	⋮	⋮	⋮
245	Node 1	Node 0	Node 1
246	Node 1	Node 0	Node 1
247	Node 1	Node 3	Node 1
248	Node 1	Node 3	Node 1
⋮	⋮	⋮	⋮
279	Node 1	Node 3	Node 1
281	Node 1	Node 3	Node 1
284	Node 1	Node 0	Node 1
288	Node 1	Node 0	Node 1
290	Node 1	Node 0	Node 1
295	Node 1	Node 0	Node 1
300	Node 1	Node 0	Node 1

The present simulations focus on the behaviour of the distributed leader-election mechanism. Therefore, the reported leader-selection time, penalty evolution and energy trends should not be interpreted as an end-to-end performance characterisation of the complete FPGA-Edge-BFT architecture.

The simulations reported here evaluate leader-election behaviour under the configured communication and recovery conditions. They do not constitute an exhaustive experimental evaluation against all Byzantine behaviours, such as equivocation, coordinated malicious participants or deliberate falsification of sensor data.

5.2. Machine Learning Training

At the same time, various databases relating to weather conditions in the metropolitan area and PM10 statistics are compiled from one of the existing air quality monitoring stations. In both cases, the same station is selected, namely the one located in O'Higgins Park (Santiago Municipality); the reason for choosing this monitoring station is the amount of information available for the training and its proximity to the university's facilities. The current weather station represents a single point of failure, as it is the only active sensor in the area; any structural or connectivity disruption renders local early-warning systems inoperable. To mitigate this operational risk and ensure the accuracy of information during crisis situations, it is necessary to deploy nodes with a Byzantine fault-tolerant architecture.

A dataset containing PM10 values for three years (2019, 2020 and 2021) is downloaded from the official SINCA website; in addition, a dataset detailing average ambient temperature and precipitation for the same period is downloaded from the DGAC website. The aim is to cross-reference this information, using ML tools to obtain a prediction of PM10 particle behaviour based on the climatological variables (the time series do not contain any missing data).

- Training period: from "2019-01-31" to "2021-05-24".
- Test period: from "2021-05-25" to "2021-12-24".

Specifically, all models execute a one-step-ahead daily forecast ($PM_{10,t}$), updating the autoregressive lags ($PM_{10,t-1}$ through $PM_{10,t-14}$) at each time step with the actual observed concentrations and incorporating the previous day's 24-h cumulative precipitation ($Rainfall_{t-1}$) to prevent data leakage, while ambient temperature was excluded during feature selection due to statistical non-significance; after removing the initial 14 observations required to construct the $PM_{10,t-14}$ lag and 36 incomplete records arising from cross-referencing the SINCA and DGAC series across the 845-day training window, exactly 795 valid observations (788 degrees of freedom) remained for model estimation.

Figure 11 shows that, during the test period, the predictions fit reasonably well with the observed PM10 values. The best-performing models, particularly XGBoost and linear regression, follow the central trend of the time series and exhibit lower error dispersion. However, there is a tendency to smooth out local peaks, suggesting that the models adequately capture the temporal persistence of the pollutant but are less capable of reproducing abrupt episodes of high concentration. This limitation is particularly relevant for high-concentration episodes, which are critical for air-quality management. Therefore, the aggregate regression metrics reported in this study should not be interpreted as a direct assessment of alert-detection capability.

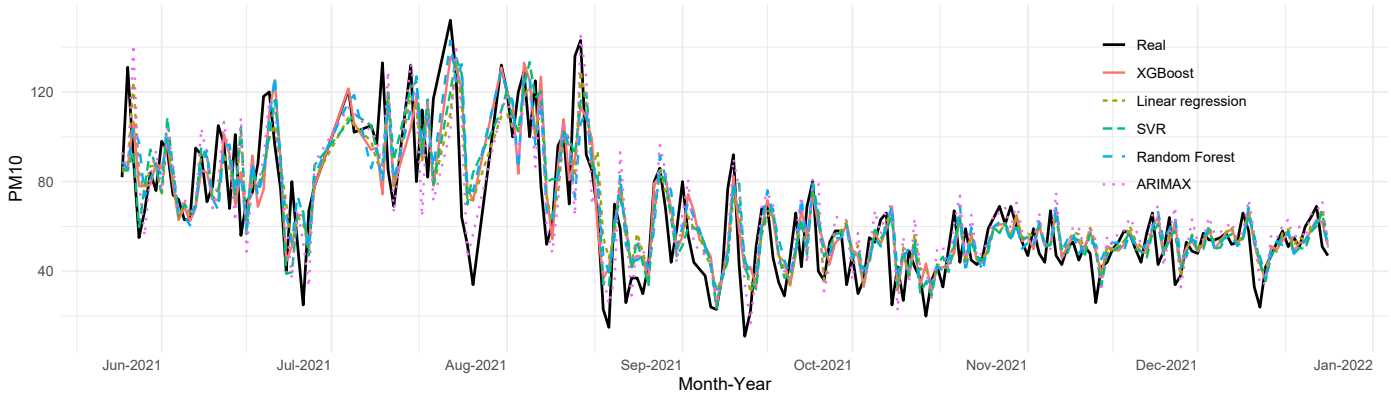


Figure 11. Comparison of PM10 forecast models during the test period (from 25 May 2021 to 24 December 2021).

No general normalisation was applied to the dataset prior to training. Tree-based models, such as Random Forest and XGBoost, were trained using the variables on their original scale, as they do not require prior standardisation. In the case of the radial kernel SVR, the predictors were normalised internally using the argument scale = TRUE, as this type of model is sensitive to the scale of the variables. In the XGBoost model, no prior standardisation of the variables was applied, as decision tree-based models do not depend directly on Euclidean distances or scalar products between variables. Therefore, the predictors were used in their original units: PM10 lags and 24-h cumulative precipitation.

For the chronological hold-out period considered in this study, XGBoost obtained the best point estimates among the evaluated models, which obtained the lowest RMSE, the lowest MAE, the lowest MSE, and the highest R^2 of the evaluated models. This model explains approximately 59.4% of the variability in PM10, with an average absolute error of 13.28 PM10 units. Therefore, after taking first place in the ranking, closely followed by linear regression, come SVR and Random Forest. The ARIMAX model improves the temporal structure but does not outperform the supervised models with lags (see Table 4 and Figure 12). The small observed differences should be interpreted cautiously, as statistical significance testing and repeated temporal validation are beyond the scope of the present comparison.

Table 4. Ranking of PM10 forecast models (25 May 2021 to 24 December 2021).

Rank	Model	MAE	RMSE	MSE	MAPE	R^2
1	XGBoost	13.284	17.681	312.623	26.418%	0.594
2	Linear regression	13.545	17.868	319.268	27.188%	0.586
3	SVR	13.646	18.099	327.587	26.643%	0.575
4	Random Forest	13.758	18.502	342.315	27.394%	0.556
5	ARIMAX	15.112	19.974	398.961	28.630%	0.482

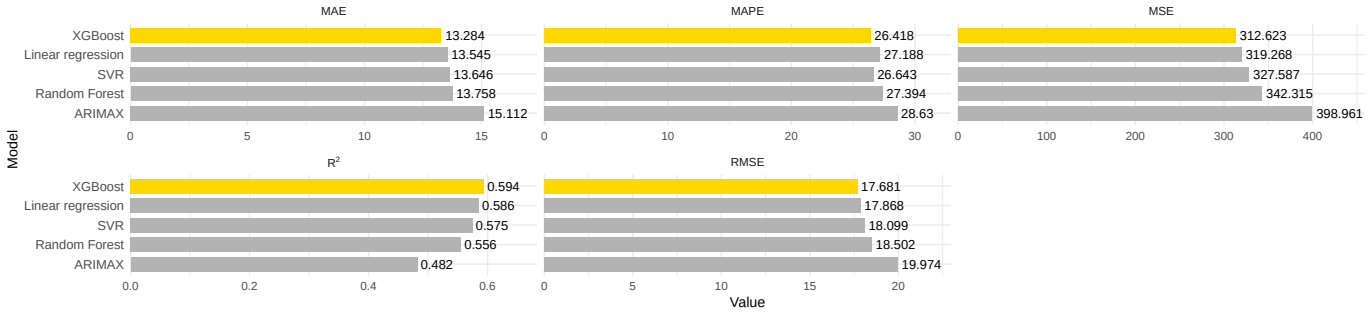


Figure 12. Metrics for PM10 forecast models (25 May 2021 to 24 December 2021).

Table 5 of coefficients shows that all predictors included in the linear regression were statistically significant (indicating that there is statistical evidence that its effect on the dependent variable is not zero, holding all other variables constant). The coefficient for $PM10_{t-1}$ is the one with the largest positive value, confirming that the previous day's pollution level is the main predictor of the current PM10. The lags $PM10_{t-3}$, $PM10_{t-7}$ and $PM10_{t-14}$ also exhibit positive effects, indicating short- and medium-term persistence. In contrast, $PM10_{t-2}$ has a negative coefficient, which can be interpreted as an adjustment effect once the strong influence of the immediate lag has been controlled for. Finally, 24-h cumulative rainfall has a significant negative effect, indicating that higher levels of rainfall are associated with lower expected concentrations of PM10.

Table 5. Linear regression model. Residual standard error: 20.42 on 788 degrees of freedom, Multiple R-squared: 0.53, Adjusted R-squared: 0.5264, F-statistic: 148.1 on 6 and 788 DF, p-value: $<2.2 \times 10^{-16}$.

Coefficients	Estimate	Std. Error	t Value	p-Value
Intercept	8.73037	2.52884	3.452	0.000585
PM10_lag1	0.67056	0.03414	19.639	$<2 \times 10^{-16}$
PM10_lag2	−0.17095	0.04090	−4.180	3.24×10^{-5}
PM10_lag3	0.13710	0.03405	4.026	6.21×10^{-5}
PM10_lag7	0.09698	0.02699	3.594	0.000346
PM10_lag14	0.15112	0.02782	5.432	7.44×10^{-8}
24-hourRainfall	−1.66839	0.25763	−6.476	1.66×10^{-10}

Table 6 shows the hyperparameters of the models. In the analysis of the models, the selection of hyperparameters plays a crucial role in the quality of the results. The ARIMA(4,0,1) error regression structure characterises the ARIMAX model, whilst the settings for num.trees (500) and mtry (3) in Random Forest suggest high complexity and an ability to handle multidimensional data. Furthermore, setting min.node.size to 5 and num.trees to 300 indicates greater stability and accuracy in the predictions. On the other hand, the choice of the reg:squarederror objective and the optimisation of parameters such as eta (0.05), max_depth (3) and subsample/colsample_bytree (0.8) in XGBoost results in high prediction accuracy. In the case of SVR, the selection of the radial kernel and the setting of cost to 10, gamma to 0.05 and epsilon to 0.01 suggest a focus on pattern recognition within the dataset. Hyperparameter selection combined automated procedures with manual specifications. For ARIMAX models, the auto.arima() function was used, which selects the model structure based on information criteria. In the Random Forest and XGBoost models, the hyperparameters were set manually, using conservative settings designed to avoid overfitting. For the SVR model, however, a grid search was applied to the hyperparameters γ, ε and C, using an internal temporal split for training and validation. This latter strategy enabled the selection of the combination with the lowest predictive error whilst respecting the chronological structure of the time series.

Table 6. Hyperparameters of the models.

Model	Hyperparameter	Value
ARIMAX	Structure	Regression with ARIMA(4,0,1) errors
Random Forest	num.trees	500
Random Forest	mtry	3
Random Forest	min.node.size	5
XGBoost	objective	reg:squarederror
XGBoost	eta	0.05
XGBoost	max_depth	3
XGBoost	subsample	0.8
XGBoost	colsample_bytree	0.8
XGBoost	min_child_weight	1
XGBoost	nrounds	300
SVR	kernel	radial
SVR	cost	10
SVR	gamma	0.05
SVR	epsilon	0.01

Model comparison shows that XGBoost achieved the best predictive performance, with MAE = 13.28, RMSE = 17.68, MSE = 312.62, MAPE = 26,42% y $R^2 = 0.594$. Linear regression performed very well, with RMSE = 17,87 y and $R^2 = 0.586$, confirming that much of the PM10 dynamics is explained by its lags and precipitation. In terms of variable importance, the one-day lag of PM10 was the dominant predictor in all models, followed by short-term and bi-weekly lags. Emissions showed a consistent negative effect, reducing expected PM10 levels.

6. Discussion

The results indicate that integrating a Byzantine Fault Tolerant distributed communication protocol with predictive ML techniques can outline a robust infrastructure for monitoring PM10 in dispersed WSN. It is worth noting that the improved performance of XGBoost, closely followed by linear regression and SVR, suggests that moderately structured, nonlinear forecasting models are effective in capturing the temporal meteorological correlations governing PM10 fluctuations. The suboptimal performance of the ARIMAX model highlights the inherent limitations of traditional time series approaches when addressing complex external influences and nonstationary behaviour.

In terms of real-time monitoring and forecasting, as presented by Ramadan et al. [77], systems have been developed that use advanced technology to detect a wide range of pollutants and provide accurate alerts. This work enables a more in-depth study of air quality in heavily congested areas by utilising additional measurement parameters and techniques such as federated learning, blockchain and energy-efficient artificial intelligence processing [78–80].

The results suggest that XGBoost has predictive capability for estimating PM10 concentrations in the specific context of O'Higgins Park; however, its application to other urban areas should be approached with caution. Any discussion of these findings must take into account that the model's performance depends not only on its architecture, but also on the spatial representativeness of the training data and the relationship between meteorological variables, emission sources and local territorial characteristics. Therefore, before using the model as a tool to support air quality management at a regional scale, it is essential to validate its performance at stations with contrasting environmental and emission profiles.

Although the current simulations focus on communication and recovery conditions rather than explicit attacks on data integrity, the proposed architecture aims to address corrupted or intentionally falsified PM10 readings through measurement redundancy, distributed data fusion, and consensus-based coordination. Specifically, multiple sensor nodes provide redundant observations of the monitored environment, enabling the system to compare measurements from geographically related sources, identify inconsistent values, and mitigate the influence of an anomalous or compromised isolated node. Furthermore, the BFT-oriented coordination layer prevents a single node from unilaterally determining the information sent to the edge layer, as distributed decisions rely on validation and agreement among a sufficient set of well-connected nodes. However, the effectiveness of this approach against explicit Byzantine-style data attacks (such as message equivocation, replay attacks, malicious node collusion, sensor drift, and deliberate falsification of PM10 payload data) has not yet been quantitatively evaluated.

The simulations reported in this study are limited to 10, 13 and 15 nodes and are intended to provide an initial assessment of leader-election behaviour rather than a large-scale scalability evaluation. In particular, as the number of participating nodes increases, so does the number of messages exchanged during the consensus phases, because the broadcast-based communication scheme requires nodes to broadcast, validate and replicate information to multiple network participants. In conventional BFT protocols, this

behaviour can result in quadratic communication complexity, of the order of $O(n^2)$, where n represents the number of nodes, which increases bandwidth usage, energy consumption, the processing load associated with message validation, and the latency required to reach consensus. Because broadcast-oriented coordination can lead to communication growth of approximately $O(n^2)$, larger-scale experiments with 20, 50, 100 or more nodes are necessary to quantify message growth, latency and energy consumption for the specific DLE implementation.

6.1. Limitations

The models were trained and evaluated exclusively using historical PM10 records from O'Higgins Park; therefore, the performance obtained for XGBoost should be interpreted in the context of the specific conditions at that site and the validation period analysed. This limitation restricts the generalisation of the results to the metropolitan scale, given that PM10 concentrations are influenced by heterogeneous spatial factors, such as traffic intensity, industrial emissions, land use, topography, local ventilation, urban morphology and the presence of green spaces.

6.2. A Future Line of Work

Therefore, as a future line of work, we propose to evaluate a more efficient communication mechanism that reduces the number of messages required per consensus round, whilst maintaining the security and fault-tolerance properties of the BFT algorithm. This optimisation would improve the solution's scalability, particularly in IoT networks with a growing number of sensor nodes, and would enable a quantitative comparison of the proposed approach's performance with previous work, taking into account metrics such as the number of messages transmitted, consensus latency, computational resource consumption, bandwidth usage and resilience to Byzantine faults.

7. Conclusions

This study presents a three-tiered architectural framework—integrating IoT acquisition, Edge/FPGA computing, and Cloud analytics—that establishes a functional roadmap to bridge Byzantine Fault Tolerant (BFT) consensus with machine learning for urban PM_{10} monitoring. By defining a clear processing hierarchy, the proposed design addresses communication resilience and short-term air quality forecasting in resource-constrained environmental sensor networks. Network simulations across small-scale topologies (10–15 nodes) demonstrated baseline protocol feasibility and startup leader election convergence within 5 s, while effectively illustrating the dynamic workload distribution between leader and follower nodes under crash and message omission events.

In terms of predictive analytics, retrospective single-station forecasting at the O'Higgins Park monitoring site established that historical daily PM10 lags and observed cumulative rainfall account for the vast majority of the variance explained. XGBoost achieved the lowest overall test error (RMSE = 17.681 $\mu g/m^3$, $R^2 = 0.594$), closely followed by Linear Regression (RMSE = 17.868 $\mu g/m^3$, $R^2 = 0.586$), whereas ARIMAX yielded comparatively poorer results due to its limited capacity for capturing non-linear meteorological dynamics. The marginal numerical lead of XGBoost emphasizes that simpler linear models capture primary daily air quality trends effectively, cautioning against inferring non-linear model superiority without broader multi-site validations.

To maintain strict scientific transparency, the scope and methodological boundaries of this investigation are explicitly defined. The reported findings represent a conceptual architectural blueprint and proof-of-concept design rather than a fully deployed physical system, as physical multi-node FPGA synthesis and real-time IoT hardware integration

remain design roadmaps. Furthermore, aggregate regression metrics evaluate central trend fitting rather than operational threshold-exceedance alert precision, and communication simulations establish protocol availability without quantifying performance under active Byzantine payload falsification or large-scale deployments (>100 nodes).

Future work will focus on transitioning this architectural framework into a physical testbed. Key upcoming research milestones include synthesizing spatial data fusion logic onto physical FPGA modules, evaluating energy and latency trade-offs across Edge CPU, Edge FPGA, and Cloud-only processing topologies, and conducting closed-loop experiments to determine whether BFT-filtered sensor streams measurably improve forecast accuracy. Finally, machine learning evaluations will be expanded to multi-station regional networks using rolling-origin cross-validation, statistical significance testing (such as Diebold-Mariano tests), and threshold-exceedance classification metrics (Precision, Recall, and Lead Time) for operational health alert readiness.

Author Contributions: Conceptualization, S.M., C.F.-C., H.V.-F., C.B. and R.C.; methodology, S.M., C.F.-C., H.V.-F., C.B. and R.C.; software, S.M. and C.F.-C.; validation, C.F.-C., H.V.-F. and C.B.; formal analysis, S.M., C.F.-C., H.V.-F., C.B. and R.C.; investigation, S.M., C.F.-C., H.V.-F., C.B. and R.C.; resources, C.F.-C. and H.V.-F.; data curation, S.M. and C.F.-C.; writing—original draft preparation, S.M., C.F.-C., H.V.-F., C.B. and R.C.; writing—review and editing, S.M., C.F.-C., H.V.-F., C.B. and R.C.; visualization, S.M. and R.C.; supervision, C.F.-C. and H.V.-F.; project administration, C.F.-C.; funding acquisition, H.V.-F. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge the financial support provided by the National Agency for Research and Development (ANID), through the FONDEF IT project IT25I0195, which made this research possible.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Full datasets may be available to academic researchers following the completion of appropriate confidentiality agreements.

Acknowledgments: The authors wish to thank the Telematic Research Projects group, part of the Department of Electrical Engineering at the University of Santiago de Chile, for the valuable support and resources provided during the course of this work.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BFT	Byzantine Fault Tolerance
DLE	Distributed Leader Election
FPGA	Field-Programmable Gate Array
IoT	Internet of Things
ML	Machine Learning
RBF	Gaussian function of a radial base
SINCA	National Air Quality Information System
SVR	Support Vector Regression
WSN	Wireless Sensor Networks

References

1. Pourghebleh, B.; Hekmati, N.; Davoudnia, Z.; Sadeghi, M. A roadmap towards energy-efficient data fusion methods in the Internet of Things. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6959. [CrossRef]
2. Sharma, S.K.; Wang, X. Live Data Analytics with Collaborative Edge and Cloud Processing in Wireless IoT Networks. *IEEE Access* **2017**, *5*, 4621–4635. [CrossRef]

3. Yassine, A.; Singh, S.; Hossain, M.S.; Muhammad, G. IoT big data analytics for smart homes with fog and cloud computing. *Future Gener. Comput. Syst.* **2019**, *91*, 563–573. [CrossRef]
4. Tu, J.; Yang, L.; Cao, J. Distributed Machine Learning in Edge Computing: Challenges, Solutions and Future Directions. *ACM Comput. Surv.* **2025**, *57*, 1–37. [CrossRef]
5. Iqbal, U.; Massoud, A.; Noureldin, A. A Sensor-Centric Survey of Autonomous Driving: Integrating Measurement Physics, Uncertainty Modeling, and Safety-Critical Multi-Sensor Fusion. *Sensors* **2026**, *26*, 3801. [CrossRef] [PubMed]
6. Strobel, V.; Castelló Ferrer, E.; Dorigo, M. Managing Byzantine Robots via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; AAMAS '18, pp. 541–549. [CrossRef]
7. Alvarado-Alcon, F.J.; Asorey-Cacheda, R.; Garcia-Haro, J.; García, L.; Garcia-Sanchez, A.J. Citywide Air Quality Forecasting over Sparse Sensor Networks: Cross-Location Generalization and Deep Learning Reliability Under Missing Data. *J. Sens. Actuator Netw.* **2026**, *15*, 52. [CrossRef]
8. Alani, N.H.S.; Chand, P.; Al-Rawi, M. A Two-Stage Machine Learning Framework for Air Quality Prediction in Hamilton, New Zealand. *Environments* **2025**, *12*, 336. [CrossRef]
9. Lee, J.; Bagheri, B.; Kao, H.A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *SME Manuf. Lett.* **2014**, *3*, 18–23. [CrossRef]
10. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [CrossRef]
11. Iskandarani, M.Z. Effect of Intelligent Reflecting Surface on WSN Communication with Access Points Configuration. *IEEE Access* **2025**, *13*, 13380–13394. [CrossRef]
12. Betis, G.; Larios, V.M.; Petri, D.; Wu, X.; Deacon, A.; Hayar, A. The ieee smart cities initiative—Accelerating the smartification process for the 21st century cities [point of view]. *Proc. IEEE* **2018**, *106*, 507–512. [CrossRef]
13. Sang, S.; Chu, C.; Zhang, T.; Chen, H.; Yang, X. The global burden of disease attributable to ambient fine particulate matter in 204 countries and territories, 1990–2019: A systematic analysis of the Global Burden of Disease Study 2019. *Ecotoxicol. Environ. Saf.* **2022**, *238*, 113588. [CrossRef] [PubMed]
14. Nyayapathi, P.P.; Namuduri, S.; Kolli, S.K. A comprehensive review of vertical profiling of ambient air quality-particulate matter and its impacts on climatic & environmental health. *Air Qual. Atmos. Health* **2025**, *18*, 1569–1595. [CrossRef]
15. Pope, C.A.; Burnett, R.T.; Thun, M.J.; Calle, E.E.; Krewski, D.; Ito, K.; Thurston, G.D. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. *JAMA* **2002**, *287*, 1132–1141. [CrossRef] [PubMed]
16. Hogea, S.; Tudorache, E.; Fildan, A.P.; Fira-Mladinescu, O.; Marc, M.; Oancea, C. Risk factors of chronic obstructive pulmonary disease exacerbations. *Clin. Respir. J.* **2020**, *14*, 183–197. [CrossRef] [PubMed]
17. Ivan, P.; Schwarz, M.; Mikušová, M. Assessment of Health Risks Associated with PM10 and PM2.5 Air Pollution in the City of Zvolen and Comparison with Selected Cities in the Slovak Republic. *Environments* **2025**, *12*, 212. [CrossRef]
18. Ministerio de Medio Ambiente. Sistema de Información Nacional de Calidad del Aire (SINCA). 2025. Available online: https://sinca.mma.gob.cl (accessed on 6 June 2025).
19. Salini Calderón, G.; Pérez Jara, P. Estudio de series temporales de contaminación ambiental mediante técnicas de redes neuronales artificiales. *Ingeniare. Rev. Chil. De Ing.* **2006**, *14*, 284–290. [CrossRef]
20. Lamport, L. Distributed Systems: A Historical Perspective. In *Distributed Operating Systems*; Tanenbaum, A.S., Ed.; Prentice-Hall: Englewood Cliffs, NJ, USA, 1987; pp. 1–20.
21. Laprie, J.C. Dependability: A Unifying Concept for Reliable, Safe, Secure Computing. In *Algorithms, Software, Architecture—Information Processing '92, Proceedings of the IFIP 12th World Computer Congress, Madrid, Spain, 7–11 September 1992*; van Leeuwen, J., Ed.; North-Holland: Amsterdam, The Netherlands, 1992; Volume A-12, pp. 585–593.
22. Weiser, M. Some computer science issues in ubiquitous computing. *Commun. ACM* **1993**, *36*, 75–84. [CrossRef]
23. Lyytinen, K.; Yoo, Y. Issues and Challenges in Ubiquitous Computing. *Commun. ACM* **2002**, *45*, 62–65. [CrossRef]
24. Satyanarayanan, M. Pervasive Computing: Vision and Challenges. *IEEE Pers. Commun.* **2001**, *8*, 10–17. [CrossRef]
25. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless Sensor Network Survey. *Comput. Netw.* **2008**, *52*, 2292–2330. [CrossRef]
26. Kandris, D.; Nakas, C.T.; Vomvas, D.; Koulouras, G. Applications of Wireless Sensor Networks: An Up-to-Date Survey. *Appl. Syst. Innov.* **2020**, *3*, 14. [CrossRef]
27. Jalili, A.; Alzubi, J.A.; Rezaei, R.; Webber, J.L.; Fernández-Campusano, C.; Gheisari, M.; Amin, R.; Mehbodniya, A. Markov chain-based analysis and fault tolerance technique for enhancing chain-based routing in WSNs. *Concurr. Comput. Pract. Exp.* **2024**, *36*, e8032. [CrossRef]
28. Gheisari, M.; Shafi, J.; Kosari, S.; Amanabadi, S.; Mehdizadeh, S.; Campusano, C.F.; Abdalla, H.B. Development of improved deep learning models for multi-step ahead forecasting of daily river water temperature. *Eng. Appl. Comput. Fluid Mech.* **2025**, *19*, 2450477. [CrossRef]
29. Lamport, L.; Shostak, R.; Pease, M. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* **1982**, *4*, 382–401. [CrossRef]

30. Distler, T. Byzantine Fault-tolerant State-machine Replication from a Systems Perspective. *ACM Comput. Surv.* **2021**, *54*, 1–38. [CrossRef]
31. Misra, A.; Kshemkalyani, A.D. Causal Ordering in the Presence of Byzantine Processes. In Proceedings of the 2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS), Nanjing, China, 10–12 January 2023; pp. 130–138. [CrossRef]
32. Gartner, F.C. Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. *ACM Comput. Surv.* **1999**, *31*, 1–26. [CrossRef]
33. Pease, M.; Shostak, R.; Lamport, L. Reaching Agreement in the Presence of Faults. *J. ACM* **1980**, *27*, 228–234. [CrossRef]
34. Fischer, M.J. The Consensus Problem in Unreliable Distributed Systems (a Brief Survey). In *Foundations of Computation Theory. FCT 1983*; Lecture Notes in Computer Science; Karpinski, M., Ed.; Springer: Berlin/Heidelberg, Germany, 1983; Volume 158.
35. Lamport, L. Paxos Made Simple. *ACM SIGACT News (Distributed Comput. Column)* **2001**, *32*, 51–58.
36. Lynch, N.A. *Distributed Algorithms*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1996.
37. Samaan, S.S.; Jeiad, H.A. A Survey of Graph Techniques Applied in Software Defined Networking. *Iraqi J. Comput. Commun. Control Syst. Eng.* **2023**, *23*, 115–124. [CrossRef]
38. Hadzilacos, V.; Toueg, S. Fault-tolerant broadcasts and related problems. In *Distributed Systems*, 2nd ed.; Mullender, S., Ed.; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1993; pp. 97–145.
39. Cristian, F. Synchronous and asynchronous group communication. *Commun. ACM* **1996**, *39*, 88–97. [CrossRef]
40. Fischer, J.; Lynch, A.; Paterson, M.S. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM* **1985**, *32*, 374–382. [CrossRef]
41. Dolev, D.; Dwork, C.; Stockmeyer, L. On the Minimal Synchronism Needed for Distributed Consensus. *J. ACM* **1983**, *34*, 77–97.
42. Dwork, C.; Lynch, N.; Stockmeyer, L. Consensus in the Presence of Partial Synchrony. *J. ACM* **1988**, *35*, 288–323. [CrossRef]
43. Chandra, T.D.; Toueg, S. Unreliable Failure Detectors for Reliable Distributed Systems. *J. ACM* **1996**, *43*, 225–267. [CrossRef]
44. Larrea, M.; Martin, C.; Soraluze, I. Communication-Efficient Leader Election in Crash–Recovery Systems. *J. Syst. Softw.* **2011**, *84*, 2186–2195. [CrossRef]
45. Fernandez-Campusano, C.; Larrea, M.; Cortinas, R.; Raynal, M. Eventual Leader Election Despite Crash-Recovery and Omission Failures. In Proceedings of the 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC), Zhangjiajie, China, 18–20 November 2015; pp. 209–214. [CrossRef]
46. Fernández-Campusano, C. Distributed Eventual Leader Election in the Crash-Recovery and General Omission Failure Models. Ph.D. Thesis, Universidad del País Vasco, Leioa, Spain, 2019.
47. Appel, M.; Gujarati, A.; Brandenburg, B.B. A Byzantine Fault-Tolerant Key-Value Store for Safety-Critical Distributed Real-Time Systems. In Proceedings of the Workshop on Certification of Safety-Critical Software Controlled Systems (CERTS 2017), Dubrovnik, Croatia, 27 June 2017; pp. 1–6.
48. Castro, M.; Liskov, B. Practical Byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA, 22–25 February 1999; OSDI '99, pp. 173–186.
49. Speranza, A.; Caggiano, R. Quantitative assessment of meteorological variables' impact on PM10 and PM2.5 aerosol particles in Rome (2015–2019): Climate change effects on PM. *Urban Clim.* **2026**, *67*, 102970. [CrossRef]
50. Karigowda; Jain, K.; Imran, M.; Un-Nabi, M.; Baig, A.; Patra, S.; Raman, R.S.; Habib, G. Particulate matter pollution and source attribution in eastern Indo-Gangetic Plains industrial corridor: Toward an airshed-based management approach. *Atmos. Environ.* **2026**, *371*, 121825. [CrossRef]
51. Vuong, Q.T.; Nguyen, T.N.T. Long-Term Trends and Machine Learning-Based Predictions of Atmospheric Visibility in the Largest Industrial City of South Korea. *CLEAN—Soil Air Water* **2026**, *54*, e70098. [CrossRef]
52. Undefinedelikten, H. Evaluating machine learning (RF, XGBoost) and statistical model (MLR) for PM10 and air quality prediction: A case from Kars, Türkiye. *Atmos. Pollut. Res.* **2026**, *17*, 102975. [CrossRef]
53. Kesten, A.S.; Panhwar, M. Modeling Particulate Matter Concentrations: A Comparative Analysis of Statistical and Machine Learning Models. *Atmosphere* **2026**, *17*, 160. [CrossRef]
54. Batina, A.; Krtalić, A. Explainable Machine Learning and Geospatial Assessment of Wildfire Smoke Impacts on Urban Air Quality in Split, Solin, and Kaštela, Croatia. *Appl. Sci.* **2026**, *16*, 6336. [CrossRef]
55. Zaman, M.; Ahnaf Zaman, M.; Chowdhury, N.A.; Jobayer Khan, M. Urban air quality shifts under lockdown restrictions: Assessing the role of policy stringency in Dhaka's COVID-19 anthropause. *City Environ. Interact.* **2026**, *30*, 100355. [CrossRef]
56. Susik, R.; Grabowski, S. Boosting exact pattern matching with extreme gradient boosting (and more). *J. Supercomput.* **2025**, *81*, 673. [CrossRef]
57. Li, Z.; Chai, H.; Lu, T.; Li, N. Modeling and prediction of regional sound speed profile grid using eXtreme Gradient Boosting model. *Phys. Scr.* **2026**, *101*, 256007. [CrossRef]
58. Liu, Z.; Lei, Q.; Weng, D.; Yang, L.; Wang, X.; Wang, Z.; Fan, M.; Wang, J. A Powerful Prediction Framework of Fracture Parameters for Hydraulic Fracturing Incorporating eXtreme Gradient Boosting and Bayesian Optimization. *Energies* **2023**, *16*, 7890. [CrossRef]

59. Garcia, E.; Calvet, L.; Carracedo, P.; Serrat, C.; Miró, P.; Peyman, M. Predictive Analyses of Traffic Level in the City of Barcelona: From ARIMA to eXtreme Gradient Boosting. *Appl. Sci.* **2024**, *14*, 4432. [CrossRef]
60. Galindo-Prieto, B.; Papanicolaou, A.; Mudway, I.S. Partial Least Squares Regression Models for Outdoor Air Pollutant Forecasting. *J. Chemom.* **2026**, *40*, e70156. [CrossRef]
61. Birinci, E.; Ekmekcioğlu, O.; Ozdemir, H.; Deniz, A. Interpretable machine learning framework for air quality prediction in Istanbul using Shapley additive explanations (SHAP). *Stoch. Environ. Res. Risk Assess.* **2026**, *40*, 37. [CrossRef]
62. Yohanna, E.; San, L.H. Comparative analysis of machine learning models for predicting PM10 and PM2.5 Concentrations in Pulau Pinang, Malaysia. *Air Qual. Atmos. Health* **2026**, *19*, 21. [CrossRef]
63. Astudillo, G.; Carrasco, R.; Fernández-Campusano, C.; Chacón, M. Copper Price Prediction Using Support Vector Regression Technique. *Appl. Sci.* **2020**, *10*, 6648. [CrossRef]
64. Bouzghiba, H.; Ajdour, A.; Omar, N.; Mendyl, A.; Géczi, G. A Novel Application of Choquet Integral for Multi-Model Fusion in Urban PM10 Forecasting. *Atmosphere* **2025**, *16*, 1274. [CrossRef]
65. Kalantari, E.; Gholami, H.; Malakooti, H.; Kaskaoutis, D.G.; Saneei, P. An integrated feature selection and machine learning framework for PM10 concentration prediction. *Atmos. Pollut. Res.* **2025**, *16*, 102456. [CrossRef]
66. Zheng, X.; Chen, Q.; Sun, M.; Zhou, Q.; Shi, H.; Zhang, X.; Xu, Y. Exploring the influence of environmental indicators and forecasting influenza incidence using ARIMAX models. *Front. Public Health* **2024**, *12*, 1441240. [CrossRef] [PubMed]
67. Tabesh, H.; AbbaszadehMozaffari, A.; Ebnehoseini, Z.; Saki, A. Enhancing the forecast accuracy of the daily number of patients arrivals in emergency department by hybrid ARIMAX-ANN algorithm. *PLoS ONE* **2026**, *21*, e0347866. [CrossRef] [PubMed]
68. Nogarotto, D.C.; Canteras, F.B.; Pozza, S.A. Brazilian truckers' strike and particulate matter (PM10) concentration: Temporal trend and time series models. *Rev. Bras. Ciências Ambient.* **2022**, *57*, 477–490. [CrossRef]
69. Castellões, T.D.O.; Rizol, P.M.S.R.; Nascimento, L.F.C. Association between Premature Birth and Air Pollutants Using Fuzzy and Adaptive Neuro-Fuzzy Inference System (ANFIS) Techniques. *Mathematics* **2024**, *12*, 2828. [CrossRef]
70. Saini, J.; Dutta, M.; Marques, G. Fuzzy Inference System Tree with Particle Swarm Optimization and Genetic Algorithm: A novel approach for PM10 forecasting. *Expert Syst. Appl.* **2021**, *183*, 115376. [CrossRef]
71. Durán, C.; Fernández-Campusano, C.; Carrasco, R.; Carrillo, E. DMLBC: Dependable machine learning for seaports using blockchain technology. *J. King Saud Univ.-Comput. Inf. Sci.* **2024**, *36*, 101918. [CrossRef]
72. Ketu, S. Spatial Air Quality Index and Air Pollutant Concentration prediction using Linear Regression based Recursive Feature Elimination with Random Forest Regression (RFERF): A case study in India. *Nat. Hazards* **2022**, *114*, 2109–2138. [CrossRef]
73. Vera, D.; Toledo, K.; Bolufé, S.; Fernández-Campusano, C. FPGAs-based Edge Computing with Distributed Leader Election on Internet of Things. In Proceedings of the 2023 South American Conference On Visible Light Communications (SACVLC), Santiago, Chile, 8–10 November 2023; pp. 118–123. [CrossRef]
74. Crain, T.; Gramoli, V.; Larrea, M.; Raynal, M. DBFT: Efficient leaderless byzantine consensus and its application to blockchains. In *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*; IEEE: Cambridge, MA, USA, 1–3 November 2018; pp. 1–8. [CrossRef]
75. Marcozzi, M.; Gemikonakli, O.; Gemikonakli, E.; Ever, E.; Mostarda, L. Availability evaluation of IoT systems with Byzantine fault-tolerance for mission-critical applications. *Internet Things* **2023**, *23*, 100889. [CrossRef]
76. Golpayegani, F.; Chen, N.; Afraz, N.; Gyamfi, E.; Malekjafarian, A.; Schäfer, D.; Krupitzer, C. Adaptation in Edge Computing: A Review on Design Principles and Research Challenges. *ACM Trans. Auton. Adapt. Syst.* **2024**, *19*, 1–43. [CrossRef]
77. Ramadan, M.N.; Ali, M.A.; Khoo, S.Y.; Alkhedher, M.; Alherbawi, M. Real-time IoT-powered AI system for monitoring and forecasting of air pollution in industrial environment. *Ecotoxicol. Environ. Saf.* **2024**, *283*, 116856. [CrossRef] [PubMed]
78. Abimannan, S.; El-Alfy, E.S.M.; Hussain, S.; Chang, Y.S.; Shukla, S.; Satheesh, D.; Breslin, J.G. Towards Federated Learning and Multi-Access Edge Computing for Air Quality Monitoring: Literature Review and Assessment. *Sustainability* **2023**, *15*, 13951. [CrossRef]
79. Chinnappan, B.; Hakim, K.; Kumar, N.S.; Elumalai, V. Blockchain and IoT integration for secure short-term and long-term air quality monitoring system using optimized neural network. *Environ. Sci. Pollut. Res.* **2024**, *31*, 39372–39387. [CrossRef] [PubMed]
80. Aref Ahmad, A.Y.; Alhulayil, M. Analysis of Energy Efficiency in Public Buildings Using AI. In Proceedings of the 2025 International Conference on Business Intelligence for Technology Innovation (ICBITI), Dubai, United Arab Emirates, 16–18 April 2025; pp. 1–8. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.