LivingFog: In-Situ Environmental Data Processing for Urgent Event Detection and Analysis

Matthieu Nicolas[0000−0002−0054−7984]
Elsa Mühl[0000−0003−3229−9126]
Christoff Andermann[0000−0003−0921−8455]
Guillaume Pierre[0000−0003−1418−8174]

Abstract Monitoring the natural environment requires collecting and processing data through a set of sensors placed in the relevant locations. To enable quick and reliable detection and analysis of events such as river floods and heat waves, fog computing technologies enable *in-situ* data processing and actuator activation without needing to rely on long-distance networks in remote regions. This chapter presents LivingFog, a fog computing platform designed to support the needs of environmental scientists. We first describe the challenges faced by environmental observatories and highlight the way fog computing addresses them. We then proceed to present LivingFog's data flow organization and the underlying system architecture. We motivate and illustrate the design and features of LivingFog with real-world scenarios from the Kaligandaki Observatory, a geohydrological observatory set in Nepal.

1 Introduction

Nature is complicated, and it can occasionally be dangerous. For instance, a seemingly peaceful river may experience a large enough flood to threaten the local human population within minutes, either in urban environments [13] or in remote locations [27]. These catastrophic events often result from cascading effects where a relatively minor root cause such as a storm may result in major and hard-to-predict consequences located far from the initial incident.

To better anticipate such events, environmental scientists operate so-called *environmental observatories* dedicated to the short-term and long-term observation of a well-defined region of interest [6]. Long-term observation data may be useful for identifying trends such as the impact of climate change, or for building accurate models of

Matthieu Nicolas, Elsa Mühl and Guillaume Pierre are with Univ Rennes, Inria, CNRS, IRISA, e-mail: {matthieu.nicolas,elsa.muhl,guillaume.pierre}@irisa.fr. Christoff Andermann is with Géosciences Rennes, Université de Rennes, CNRS, e-mail: christoff.andermann@univ-rennes.fr.

the environment's response to specific events. On the other hand, short-term observation may help quickly identifying events when they occur, analyzing the detected events, and issuing timely and accurate warnings to relevant authorities.

Environmental observatories are composed of a number of sensors of various types placed in relevant locations. Typical observatories may exploit dozens to hundreds of sensors which communicate their produced data within the observatory to a local "data logger" which is in charge of buffering the data until they can be sent to a remote cloud for further analysis. Depending on local conditions, data transmission may take place periodically such as once per day. Although this is a perfectly suitable method for long-term data collection, a different approach is necessary to handle real-time data analysis. To enable urgent event detection and analysis, a promising approach proposes to enhance the environmental observatories using fog computing technologies to process sensor data locally within the observatory itself [16].

Designing a fog computing platform according to the specific requirements of *in-situ* environmental data processing requires one to address a number of difficult issues. Observatories may be located in areas where cellular networks and electricity supply are limited. Sensors may fail in a variety of ways and produce bogus data that must be filtered before further processing. In case of any detected dysfunction, the platform must generate alerts so observatory managers can quickly identify and fix the issue. Finally, an observatory may need to detect and analyze numerous different types of events, requiring the ability to deploy many independent applications to process overlapping subsets of the collected data in real time.

This chapter discusses the architecture of the LivingFog platform. LivingFog was originally designed to handle smart city scenarios [3]. We redesigned parts of the system architecture to adapt it to the specific requirements of natural environment observation for urgent event detection and analysis. We describe the overall system architecture and discuss how this architecture is designed to handle the specific case of the Kaligandaki observatory, a geohydrological observatory set in Nepal.

The remainder of this chapter is organized as follows. Section 2 presents background information about environmental observatories, then Section 3 presents the design of the LivingFog platform. Section 4 discusses the current limits and future directions. Finally, Section 5 discusses the related work and Section 6 draws conclusions.

2 Environmental observatories and their current limits

Environmental observatories are used to characterize the nature's behavior in a given geographical area. They enable one to monitor their key metrics for short-term or long-term observation. To this end, environmental observatories collect environmental data such as temperature, CO_2 concentration, and rain intensity, using a variety of sensors.

Environmental observatories can be classified in two main categories [16]. *Long-term monitoring observatories* aim to collect metrics for monitoring purposes over extended periods of time, e.g., years or decades. They rely on a static configuration, with consistent data collection frequency and transmission frequency. On the other hand, *event-driven observatories* aim to observe specific events such as floods and heat

waves. As they are only interested in metrics during specific events, they may adapt their data collection frequency and transmission frequency accordingly.

Environmental observatories however suffer from several constraints due to the fact that they are often located in remote and isolated areas. In these cases, environmental observatories may have no access to a power grid, limited and/or intermittent Internet connection, and possibly no opportunities for physical human access during parts of the year due to weather conditions or distance.

In particular, the limited availability of reliable Internet connection has profound implications for environmental observatories, as it may prohibit the usage of traditional cloud platforms to handle time-critical tasks:

(i) Detecting and responding to environmental events in near real-time. Indeed, events may start and complete while the connection is down, leading to missed opportunities to collect data and to react to the event sufficiently early to be useful.
(ii) Monitoring and reacting to sensor malfunctions, e.g., sensors producing bogus data or no data at all. Indeed, the limited available bandwidth may prevent observatories from uploading every collected data, especially multimedia data such as audio and video.

These challenges justify the need for processing sensor data *in-situ*, within the observatory itself. The data loggers that are used to buffer sensor data before transmitting them sometimes include some data processing capabilities, but they rely on proprietary programming languages and cannot support large numbers of simultaneous data processing activities. To address these limitations, we recently proposed to exploit fog computing technologies in environmental observatories [16]. Fog computing introduces one or more layers of nodes between the edge (where data are produced), and the cloud (where long-term data may be stored and processed) [5, 34]. The introduction of processing nodes close to the sources of data enables fog computing to reduce the latency and the dependency on cloud platforms. In the context of environmental observatories, they enable *in-situ* processing to detect and react to events in near real-time. They also enable one to monitor the collected data in order to detect and alert operators in case of sensor issues.

Fog computing approaches have previously been proposed in many different contexts such as Smart Cities, Industry 4.0 and Smart Agriculture [1, 10]. These contexts however differ from the one at hand as they do not share the same limitations, especially the limited energy budget and Internet connectivity. In this work, we present LivingFog, a fog computing platform designed for environmental observatories.

3 The LivingFog platform

The LivingFog platform[1] was originally designed to answer the requirements from smart city scenarios, and in particular those of marina management [3]. In this work,

[1] Source code available at: https://gitlab.inria.fr/livingfog.

Fig. 1 LivingFog installation in Lete, Nepal.

we adapt LivingFog to the domain of environmental observatories and to the specific use case of the Kaligandaki Observatory (see Figure 1).

The Kaligandaki observatory is located in Nepal, spanning an ideal transect across the Himalayan Mountain Range from the edge of the Tibetan Plateau, through the high mountains, to the low-lying Gangetic foreland. It is composed of two major stations for climatological and hydrological monitoring dedicated to the observation of the Gandaki river [32] which regularly experiences catastrophic floods during the monsoon seasons [4].

The Kaligandaki Observatory's needs, i.e., to monitor the sensor network's health or to detect and to react to events, are however common among environmental observatories. The proposed system is thus designed to generically address these issues.

3.1 System overview

In this section, we state the features and non-functional properties that the fog platform should provide. These features and properties stem from the intended use case and users, i.e., environmental observatories and the environment scientists operating the observatory and studying the given territory.

3.1.1 Required features

Feature 1. (Ingestion of sensor data) The LivingFog platform should first be able to ingest sensor data in order to perform *in-situ* data processing and monitor the sensor network's health. However, it is not LivingFog's purpose to provide long-term storage: the platform is designed as an addition to current systems, not as an alternative to data loggers. LivingFog is therefore designed for urgent data processing or hot data processing.

Feature 2. (Deployment of custom applications) Considering that environment scientists know best the collected data and the types of processing they require, the LivingFog platform should enable them to deploy their own applications.

Feature 3. (Reactions to events) The LivingFog platform should enable its applications as well as external devices, e.g., the stations, to react to events.

Feature 4. (Visualization of data in near real-time) To enable users to grasp quickly and precisely the current state of the territory and of the system, LivingFog should provide data visualization tools. These tools would also prove useful during configuration and update tasks, e.g., to manually check that new sensor data are correctly ingested by LivingFog.

Feature 5. (Alerts in case of anomalies) The LivingFog platform should provide tools to set a variety of alerts to cover (i) environmental events, e.g., floods or heat waves; (ii) sensor malfunctions, e.g., a sensor sending no more data or sending bogus data instead; (iii) its own system malfunctions, e.g., failures of components of the system.

3.1.2 Required non-functional properties

The conditions in which fog platforms should operate inside environmental observatories also allow us to identify a number of desirable non-functional properties for the LivingFog platform.

Property 1. (Resilient) As environmental observatories may be located in a remote and isolated location, the LivingFog platform should tolerate and recover from (i) energy shortages; (ii) network interruptions; (iii) hardware failures, e.g., broken physical nodes, without requiring physical interventions.

Property 2. (Energy-wise autonomous) Because of their remote locations, environmental observatories may have no access to a power grid. Therefore, the LivingFog platform should be designed to be powered by locally-available renewable energy only, without exceeding the energy budget provided by these energy sources.

Property 3. (Accessible to non-IT professionals) Since the LivingFog platform is aimed at environmental scientists, the platform should be accessible to non-IT professionals. It should then provide a user-friendly interface to access its various features, e.g., data visualization, custom applications management or alerts management.

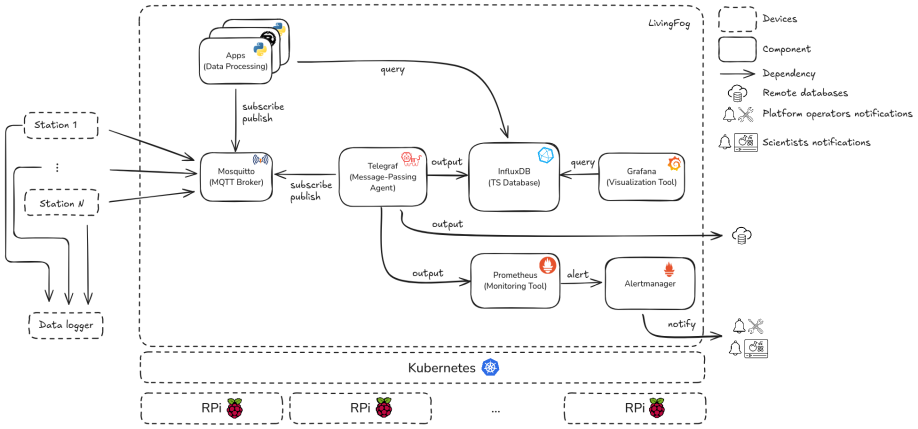

Fig. 2 System architecture of the LivingFog platform.

Property 4. (Collaborative) As the computing resources are limited in their computing capabilities as well as energy usage, it is important to pool the available resources as efficiently as possible. Thus, it is useful to share tasks, sensor data and application results between the LivingFog platform's users. To this end, LivingFog should foster the collaboration between them, for example by providing data discovery tools.

3.2 System Architecture

The LivingFog platform is designed to complement current systems without replacing existing installations such as sensors and data loggers. Sensors therefore send their data to the data loggers but also to LivingFog through its entry point: an MQTT message broker which serves as the main component to distribute data to the other LivingFog elements (see Figure 2).

Deployed applications may subscribe to the message broker's topics to retrieve in near real-time the data they are interested in. They may send their output back to the message broker as new topics which may in turn be consumed by other applications.

The message data are also read by a message-passing agent which is in charge of transmitting them to other components:

(i) A timeseries database which is in charge of storing data for a specified time window. During this period, applications can benefit from the database engine to perform complex queries on given time ranges. LivingFog also embeds a visualization tool which queries the timeseries database to build near real-time dashboards of the collected data.
(ii) A monitoring service capable of issuing alerts in case of detected anomalies stemming from environmental events or system issues.

These different components are all managed by an orchestrator, which runs on top of a cluster composed of one or more single board computers such as Raspberry Pis.

Message Broker: The MQTT message broker (based on Eclipse Mosquitto [9]) is the entry point of the LivingFog platform and its angular stone. The different components of the system share data through this mean using topics, whether it is sensors' data or applications' results. The message broker enables sensor stations to send their data to the LivingFog platform and applications to retrieve and process them in near real-time. It decouples data publishers and subscribers, enabling for example several sensors to independently feed a given topic, and applications to independently process this topic. As stations can also subscribe to topics, they are also able to react to detected events, e.g., to activate a sensor or to update its configuration when necessary.

Message-Passing Agent: The message-passing agent (based on Telegraf [12]) is in charge of forwarding messages from specified topics to the timeseries database, to the monitoring service and to remote databases and cloud platforms. Additionally, it provides features to manipulate the message content in the process, e.g., extracting tags from the name of the topic or adding new values.

Timeseries Database: In contrast to transient messages, the timeseries database (based on InfluxDB [11]) provides a permanent storage service. It enables applications to rely on its query engine to ease their tasks and retrieve data over time ranges. It also provides data to the visualization tool to show the evolutions of the monitored environment and the system itself. Given the hardware constraints of the LivingFog platform, the storage service provided can only be temporary. The timeseries database thus defines and uses data retention policies to remove obsolete data.

3.2.1 Applications

The applications constitute the user-managed components of the LivingFog platform: environment scientists should develop and deploy their own applications. Applications interact with the message broker to retrieve fresh data; they may additionally query the timeseries database to retrieve past data. This enables them to process the data *in-situ* in near real-time according to their actual needs, e.g., to validate sensor data, compute new metrics from sensor data or detect events. Once applications produce results, they are expected to publish them back to the message broker as new metrics.

All applications run in LivingFog in the form of Kubernetes pods, so they must be developed as Docker images.

3.2.2 Monitoring Service

The monitoring service (based on Prometheus [25] and its *Alertmanager* component) provides tools to monitor the different components of the system, i.e., the sensor network's and the LivingFog platform's health status. In practice, it enables the definition of *alert rules* over given metrics and the definition of *routes* to specify which users

should be notified when these rules are broken. Each *route* also enables one to describe the means which should be used to notify the users: email, sms, etc.

3.2.3 Orchestrator

The LivingFog platform's components, as well as user applications, are managed by an orchestrator (based on the lightweight k3s Kubernetes distribution [14]). The orchestrator takes care of the full life-cycle of applications from deployment to withdrawal, including network communications, interactions between the different applications, and self-healing in case containers or server devices fail.

3.2.4 Hardware

As the LivingFog platform may be constrained by a limited energy budget, we rely on single-board computers: such machines are known to provide interesting power efficiency properties [2]. We are currently using one or more Raspberry Pi 4B as physical devices for the LivingFog platform. We consider experimenting with heterogeneous clusters, as new use cases and applications are introduced, e.g., to run AI models on dedicated hardware [21, 30].

3.3 Data Flows

In this section, we describe the different data flows we implemented on top of the LivingFog system. While these data flows are tailored for our use cases at the Kaligandaki observatory, the features they provide and the problems they address are of interest for the majority of event-driven environmental observatories.

3.3.1 Cross-correlation of precipitation sensors

The first issue that environmental researchers mentioned about their existing observatory is the issue of sensor dysfunctions. In some cases the sensors simply stop producing data. In other cases, while a faulty sensor keeps producing data as expected, it actually encounters a failure and reports bogus data. In such cases it is important to detect the anomaly, report it to the environmental researchers, and filter away the bogus data so that other running applications are fed only with plausible data.

A simple approach to handle such cases is to define thresholds of acceptable values and to filter values to accept only those in a sensible range. Out-of-range data may then be reported as anomalies.

However, some sensors produce data in faulty scenarios that are harder to filter out. For example, the Vaisala WXT520 sensor [31] measures the amount of rainfall in millimeters since the last measurement. When it detects a dysfunction, instead of the

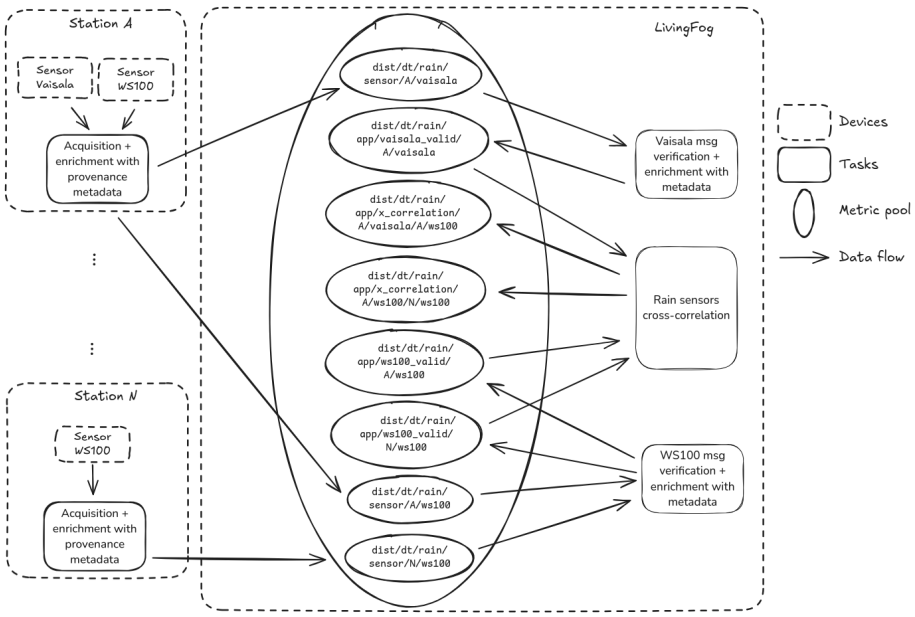

Fig. 3 Data flow to monitor and cross-correlate multiple rain gauges.

measured data it reports only `0` values. This value is at best ambiguous, as it may also express the absence of precipitation.

The approach we thus adopt to detect such faulty behaviors is to cross-correlate the measurements of the same high-level metric produced by multiple sensors. In the case of the Kaligandaki observatory, the station is also equipped with the Lufft WS100 sensor [19] which embeds a precipitation sensor, enabling this comparison. The data flow in Figure 3 illustrates this cross-correlation.

In the figure, several stations equipped with the Vaisala WXT520 and the Lufft WS100 send their collected data to the LivingFog platform. These data are streamed to the LivingFog's message broker under different topics such as `dist/dt/rain/sensor/A/vaisala`, `dist/dt/rain/sensor/A/ws100` and `dist/dt/rain/sensor/N/ws100`.

The first step to identify and filter bogus data is to verify that the messages are syntactically valid. To this end, a first application is deployed for each topic to verify that messages are in the expected format (CSV, JSON, . . .) and provide the expected schema (fields, data types, . . .). Each application also enriches the messages with LivingFog's metadata, e.g., the node identifier, the system version, etc. The syntactically-checked messages are then published in other topics: `dist/dt/rain/app/vaisala_valid/A/vaisala`, `dist/dt/rain/app/ws100_valid/A/ws100` and `dist/dt/rain/app/ws100_valid/N/ws100`.

The `Rain sensors cross-correlation` application starts from there. It defines a monitored time window and retrieves relevant measurements on a per-sensor basis. It then uses the "Student's t-test" statistical method [24] to check if the measurements of

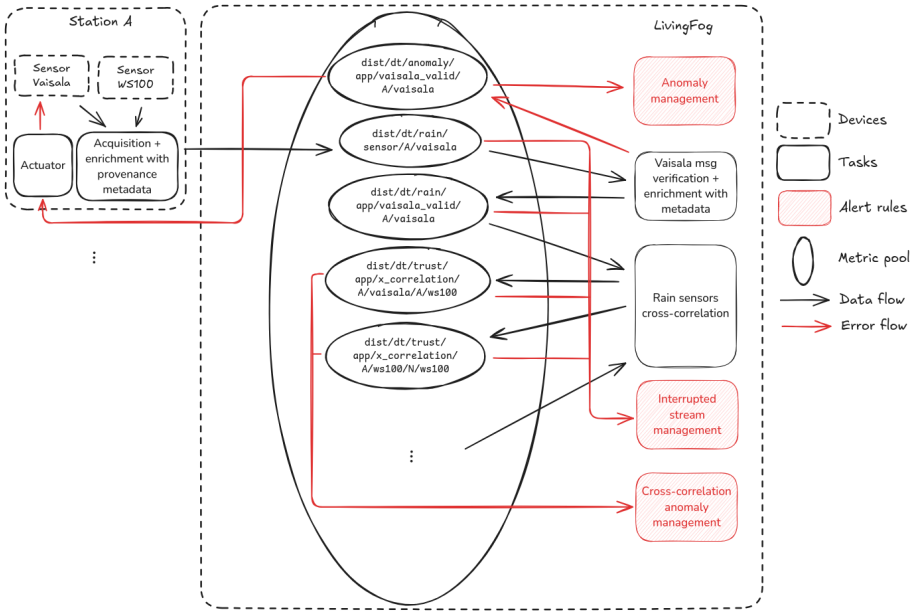

Fig. 4 Dataflow to manage sensor network issues and events of interest.

each pair of sensors are correlated. It outputs the resulting trust scores as percentages in the resulting topics `dist/dt/trust/app/x_correlation/A/vaisala/A/ws100` and `dist/dt/trust/app/x_correlation/A/ws100/N/ws100`.

This result can then be monitored to detect unsatisfying results, identify the sensor diverging from the rest, and take action. We present the monitoring data flow next.

3.3.2 Monitoring of sensor network and events

Figure 4 revisits the rain gauge processing flow while focusing on error paths and how the LivingFog platform handles them.

We first focus on the Vaisala WXT520's data. When sensor data are streamed in the LivingFog platform, the messages' formats and schemas are verified by a first application. While valid messages are forwarded to the topic `dist/dt/rain/sensor/A/vaisala`, invalid ones are instead routed to another topic: `dist/dt/anomaly/app/vaisala_valid/A/vaisala`. A peak of messages being published in this topic is therefore a good indication of a sensor issue such as a mismatch between the sensor and the application configurations, and a partial failure of one of these two components. Multiple system components may observe this topic and react to the increase of invalid messages:

(i) The station itself can subscribe to the topic. If it observes that one of its sensors is transmitting invalid data, it may for instance try to reboot the faulty sensor in a first attempt to fix the issue.
(ii) The monitoring service may also be configured using alert rules to monitor this topic. If it observes a large number of messages over a given time interval, it may be configured to notify the maintainers of the ongoing issue.

Another common case of sensor dysfunction observed is the sensor interruption, i.e., when the LivingFog system stops receiving data from a faulty sensor. This may occur if the corresponding sensor fails, if there is a mismatch between the expected output topic of the sensor and the actual one, or if the network is disconnected between the station and the LivingFog platform. Applications may also show similar issues. The *alert rules* provided by the monitoring tool feature operators to monitor the absence of messages. The LivingFog platform can thus alert the maintainers in such case, in a similar manner as previously described.

The last case described in Figure 4 concerns the topics `dist/dt/trust/app/x_correlation/A/vaisala/A/ws100` and `dist/dt/trust/app/x_correlation/A/ws100/N/ws100`. These topics carry trust scores produced by the `Rain sensors cross-correlation` application for each pair of sensors. Once again, alert rules may be defined to detect sensors having low trust scores and to issue notifications about them.

3.4 Toward urgent processing of environmental events: the example of flood detection

The next use-case we are currently working on relates to the implementation of a flood detection system. Identifying floods in a timely fashion is useful for two main reasons: (i) Issuing early warnings to the local population as floods may sometimes have catastrophic consequences; (ii) Triggering actuators that are deployed alongside the river to collect samples of the water at periodic intervals. These water samples are of particular interest to hydrogeologists because their chemical compositions provide clues about the flood's origins. In both cases, the flood detection system should issue notifications as early as possible after the beginning of a flood event.

Figure 5 presents the envisaged data flow for such use case. Detecting a flood may be done in a number of ways depending on the conditions and the list of available sensors. In the context of the Kaligandaki observatory, the station embeds a radar level sensor [22], as well cameras and seismometers. The following approaches can thus be proposed to design flood detection systems:

(i) The radar level sensor provides measurements of the height of the water surface. A simple alert rule can be implemented based on threshold values.
(ii) Cameras can be pointed toward the river banks. Image processing enables the estimation of the riverbed's width, outputting numerical values that can be used for threshold testing.

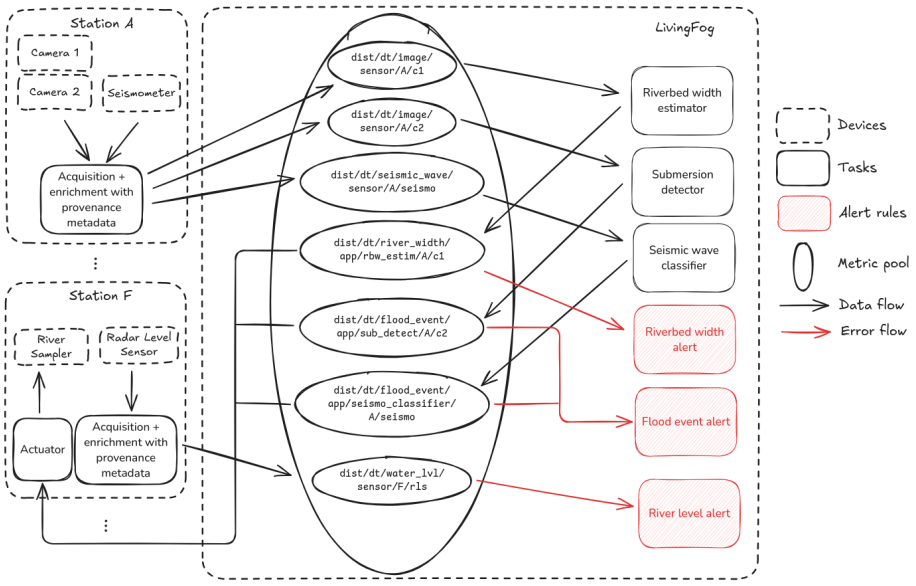

Fig. 5 Data flow to detect and react to flood events.

(iii) The cameras can also be set to monitor a reference point. Image processing enables one to determine if this reference point is submerged or not, outputting boolean values.

(iv) Ongoing research attempts to exploit seismometers to detect environmental events such as rain and floods by using classifiers. If successful, this method would enable early detection of floods occuring upstream.

Each approach provides a different tradeoff on precision, time to alert and required computing resources. For instance, the river level sensor-based method does not enable one to anticipate floods while the seismometers-based one is expected to. It is however expected to require more computing resources as it will likely need to exploit machine learning techniques to classify seismic events. Benchmarks should therefore be run to evaluate their respective precision, time to alert and impacts on the LivingFog platform's performance.

4 Discussion

The development of LivingFog is still ongoing. We now discuss the extent to which the platform is able to meet the requirements expressed by environmental researchers, and depict promising directions for future system-level research.

4.1 Meeting the environmental observatories' needs

In Section 3.1, we described the features that the LivingFog platform should provide as well as the non-functional properties it should exhibit.

The LivingFog platform's message broker depicted in Figure 2 enables stations to send their data to the system in addition to their transmission to data loggers. The data become accessible to the different components of the system: the applications, the timeseries database and the associated visualization tool, and the monitoring service. The proposed system architecture thus provides Feature 1, Feature 4 and Feature 5.

The message broker also enables applications and stations to subscribe to topics relevant for them, enabling them to process messages and to trigger reactions if necessary. We presented such a scenario in Figure 4. Feature 3 is thus also provided by the LivingFog platform.

Additionally, dedicated applications may be deployed to generate alerts (Feature 5). Combining them with Prometheus's *alert rules* enables us to overcome the limitations of Prometheus' native query language.

The LivingFog platform relies on the Kubernetes orchestrator to deploy and manage its core system-level applications. This tool is also granted to users, enabling them to deploy their own applications. The LivingFog platform thus provides Feature 2.

Kubernetes also handles application restarts if they encounter transient errors, or redeployments on other nodes if the cluster is downsized. It thus strengthens the resilience of the system, as desired with Property 1. It also provides features to automatically scale applications up or down. Applications could then be scaled up to load balance in cases of temporary overload. These features may also be leveraged to scale down or even shut down some applications in case of energy scarcity. It thus paves the way for Property 2, namely energy autonomy.

One requirement which is not fully satisfied at the moment is Property 3, i.e., the easy accessibility of the system for non-IT specialists. The LivingFog platform currently lacks a unique user-friendly interface to access and use its different services. For example, users should ideally be able to deploy an application, edit its configuration and visualize its output data with minimal knowledge of the underlying tools. Also, the requirement of providing applications as Docker images for ARM architectures is a significant barrier to the adoption of the system by environment scientists. Efforts should therefore be made toward a service easing the generation of these images, or at least supporting the environmental scientists in this task.

An interesting aspect of LivingFog is that it is inherently a multi-tenant system, with numerous environmental scientists sharing the same fog computing platform to perform different tasks based on the same set of data sources. While this fosters data sharing and reuse, as intended with Property 4, multitenancy may also generate issues in periods of resource scarcity in defining the relative priority of different applications belonging to different users. Also some users may refuse sharing data originating from certain sensors until they had the opportunity to publish their own research results first. LivingFog should thus also provide an access control system to enable users to define which other users have read or write access over certain MQTT topics. Finally, in order to maintain a fair access to its computing resources, the LivingFog platform

should implement a *user karma score* system in order to penalize those who hoard the platform's resources to themselves.

4.2 Data trust level vs. latency tradeoffs

Reliable execution of applications deployed on the LivingFog platform may be critical. For instance, in Section 3.4, we discussed the design of flood detection and early warning systems. As such events may potentially endanger human populations, the accuracy and responsiveness of the system are of paramount importance. However, the sensors which enable such detection are not foolproof and may start to provide erroneous data. Relying on unverified data may lead to false positives or false negatives, which would decrease the trust in the system.

To ensure data quality, several approaches can be adopted such as the cross-correlation technique depicted in Figure 3. However, they may introduce latency, as they require message buffering and additional computations. The applications may also fail, e.g., due to bugs. There is thus a trade-off to observe between the trust level in the input data of critical applications and latency needed to obtain them.

This also raises questions for non-critical applications. As these applications are non-critical, they might accept to rely on unverified sources of data. But in the case of an *a posteriori* detection of an anomaly in the input data, what should happen to the produced results or to the other results produced based on these tainted results? Approaches from the event stream processing literature should thus be studied to determine how to handle such cases.

4.3 Data schema validation

The intended sharing and reuse of data between multiple sensors and user applications requires that data schemas should be unambiguous and agreed upon. This mandates clear contracts between the interacting publishers and subscribers, sensors and applications alike. Senders and receivers must agree on message syntax, i.e., the format used and its embedded data, as well as message semantic, i.e., the meaning of the embedded data.

However, as the system evolves over its intended lifetime, the list of available sensors and applications will change. Each individual sensor and application will also undergo a number of bug fixes and feature improvements. These changes are error prone, especially in the case where publishers and subscribers are separate entities. These changes may lead to data deserialization issues and to applications crashes if not handled properly.

To support the natural evolutions of the fog platform, an interesting direction relates to the usage of schema registry services to maintain data schemas as first-class objects that can be published and queried [7, 26].

4.4 Energy scarcity management

A majority of environmental observatories need to operate in remote location where the only available energy sources are renewables such as solar panels attached to batteries. This creates strong fluctuations in energy availability for the fog platform: solar panels do not produce energy during the night, and are prone to large production variations due to weather and season. The LivingFog platform should therefore be able to adapt its execution to adjust energy usage to the conditions at hand. Depending on the current energy budget, some applications may need to be temporarily interrupted or slowed down, and some compute nodes may need to be temporarily turned off.

The first measure to handle this requirement is to rely on energy-efficient hardware. Single-board computers such as Raspberry Pis and edge AI devices provide an interesting tradeoff between compute capabilities and energy draw [17]. The usage of multiple such devices enables the delivery of potentially large amounts of compute power while providing opportunities to switch parts of the platform off to save energy when necessary [15].

Another approach is to define multiple criticality levels for applications. The orchestrator should then be configured to prioritize critical applications and theirs dependencies, i.e., the applications producing the data they use as inputs. Other applications may be stopped if necessary.

Finally, some applications may be able to expose multiple modes of operation: an optimal one which delivers the full capabilities of the application but potentially requires significant compute resources, and one or more degraded modes which reduce output quality but require fewer resources. For instance, an application which analyzes a live video feed may switch to lower image resolution or frame rate. This approach, which is often referred to as transprecision [20], can be applied in the context of continuous data streams [23]. In times of energy scarcity, LivingFog should be able to switch to the degraded modes to reduce energy consumption.

5 Related Work

Using computer systems to process sensor data for monitoring natural phenomena has a long and rich history. For instance, agricultural domains such as precision beekeeping introduces sensors in beehives to monitor their state, aiming to minimize resource consumption and maximize the productivity of bees [10, 35]. Precision beekeeping shares a number of challenges with environmental observatories: smart beehives are often placed in remote locations with limited or no Internet connection, and they need to operate on a limited energy budget. This domain thus shares research questions such as how to reduce network usage using *in-situ* computations to allow the usage of LPWAN networking [29]. The main difference with environmental observatories relates to the role of users: in precision beekeeping, beekeepers and researchers are customers of the provided services. On the other hand, LivingFog is designed as a Platform-as-a-Service

environment for numerous environmental researchers, enabling each of them to deploy their own applications.

Another related initiative is CEBA ("Environmental Cloud for the Benefit of Agriculture"), a data lake dedicated to sharing and exploiting environmental and agricultural scientific data [28]. As such, it collect heterogeneous data from a variety of environmental sensors. The CEBA and LivingFog platforms share common features including the ingestion, processing, visualization and querying of heterogeneous data streams. They both adopt a data-flow-based architecture to provide these features. They however differ in their respective scopes: CEBA mostly aims to provide long-term data archival and indexing. Data providers are therefore encouraged to follow the Findable, Accessible, Interoperable, and Reusable (FAIR) [33] and INSPIRE [8] principles. On the other hand, the LivingFog platform focuses on providing *in-situ* processing capabilities of data with only short-term archival. It focuses on the urgent processing of environmental data and aims to complement rather than replace cloud-based platforms such as CEBA.

Scenarios where environmental data processing must be realized without access to a stable energy supply often use Intermittent Computing approaches [18]. Intermittent systems are devices which harvest energy from their environment, e.g., from sunbeams, radio waves or vibrations. Devices operate only after they have accumulated enough energy, until they deplete their energy budget and return to the harvesting state. This mode of operation enables the usage of extremely tight energy budgets. On the other hand, data processing execution frequently needs to interrupt for prolonged periods of time because of power failures. Intermittent computing is therefore not suitable for scenarios such as urgent environmental event detection and analysis where low detection and processing delays are essential.

6 Conclusion and Future Work

In this work, we have highlighted the constraints faced by environmental observatories. In particular, environmental observatories are characterized by their unreliable Internet connectivity, absence of stable and unlimited energy supply, and requirements for urgent data processing in order to detect and analyze environmental events such as floods and heatwaves.

To address this set of characteristics, we propose the usage of fog computing technologies which enable *in-situ* processing of sensor data. Fog platforms may execute using energy-efficient single-board computers such as Raspberry Pis which provide an interesting tradeoff between computing capabilities and energy usage.

We described the LivingFog platform which was specifically designed to handle the requirements of natural environment observation. LivingFog is meant to store and process sensor data within a relatively short time window after they have been produced. It is organized along a data flow-based architecture which may be leveraged to answer to environmental observatories' needs, e.g., the validation of sensor data and the monitoring of the sensor network health and of environmental events.

The development of the LivingFog platform is still work in progress. A first operational version was deployed in November 2025 in the Kaligandaki observatory. We

intend to keep enriching it with additional features as they get developed. In particular, the current system still lacks a number of useful features for environmental researchers such as a user-friendly interface to develop and package applications. There also remains open system-level questions such as the design of energy-aware application scheduling system. We leave these topics for future work.

Acknowledgments

This research was supported by funding from the French government, administered by the National Agency for Research (ANR) through the Future Investment Program, integrated into France 2030, with grant reference ANR-21-ESRE-0014.

References

1. Ahmed, A., Arkian, H., Battulga, D., Fahs, A.J., Farhadi, M., Giouroukis, D., Gougeon, A., Gutierrez, F.O., Pierre, G., Souza Jr., P.S., Tamiru, M.A., Wu, L.: Fog computing applications: Taxonomy and requirements. CoRR **abs/1907.11621** (2019). URL http://arxiv.org/abs/1907.11621
2. Basford, P.J., Johnston, S.J., Perkins, C.S., Garnock-Jones, T., Tso, F.P., Pezaros, D., Mullins, R.D., Yoneki, E., Singer, J., Cox, S.J.: Performance analysis of single board computer clusters. Future Generation Computer Systems **102** (2020). DOI 10.1016/j.future.2019.07.040
3. Battulga, D., Farhadi, M., Tamiru, M.A., Wu, L., Pierre, G.: LivingFog: Leveraging fog computing and LoRaWAN technologies for smart marina management (experience paper). In: Proc. ICIN (2022). DOI 10.1109/ICIN53892.2022.9758124
4. Bell, R., Fort, M., Götz, J., Bernsteiner, H., Andermann, C., Etzlstorfer, J., Posch, E., Gurung, N., Gurung, S.: Major geomorphic events and natural hazards during monsoonal precipitation 2018 in the Kali Gandaki Valley, Nepal Himalaya. Geomorphology **372** (2021). DOI 10.1016/j.geomorph.2020.107451
5. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proc. MCC Workshop on Mobile Cloud Computing (2012). DOI 10.1145/2342509.2342513
6. Brantley, S.L., McDowell, W.H., Dietrich, W.E., White, T.S., Kumar, P., Anderson, S.P., Chorover, J., Lohse, K.A., Bales, R.C., Richter, D.D., Grant, G., Gaillardet, J.: Designing a network of critical zone observatories to explore the living skin of the terrestrial Earth. Earth Surface Dynamics **5**(4) (2017). DOI 10.5194/esurf-5-841-2017
7. Confluent: Confluent schema registry for Kafka. URL https://github.com/confluentinc/schema-registry. Last Accessed: 2025-08-29
8. Drafting Team Metadata and European Commission Joint Research Centre: INSPIRE metadata implementing rules: Technical guidelines based on EN ISO 19115 and EN ISO 19119 (2013). DOI 10.25607/OBP-1966
9. Eclipse Foundation: Eclipse Mosquitto. URL https://mosquitto.org/. Last Accessed: 2025-06-24
10. Hadjur, H., Ammar, D., Lefèvre, L.: Toward an intelligent and efficient beehive: A survey of precision beekeeping systems and services. Computers and Electronics in Agriculture **192** (2022). DOI 10.1016/j.compag.2021.106604
11. InfluxData Inc.: InfluxDB 3 core | InfluxData. URL https://www.influxdata.com/products/influxdb/. Last Accessed: 2025-06-24
12. InfluxData Inc.: Telegraf. URL https://www.influxdata.com/products/telegraf/. Last Accessed: 2025-06-24

13. itvNEWS: Spain's deadliest floods in decades: Death toll reaches 205 as temporary morgue opens (2024). https://www.itv.com/news/2024-11-01/spain-floods-rescuers-search-for-missing-as-death-toll-hits-158
14. K3s Authors: K3s. URL https://k3s.io/. Last Accessed: 2025-06-25
15. Kazem, A., Pierre, G., Longuevergne, L.: Demo: Energy-aware dynamic dimensioning of IoT edge clusters for natural environment observation. In: Proc. IEEE IC2E (2025). URL https://hal.science/hal-05218930
16. Kazem, A., Pierre, G., Longuevergne, L.: Enhancing environmental observatories with fog computing. Frontiers in Environmental Science **13** (2025). DOI 10.3389/fenvs.2025.1568016
17. van Kempen, A., Crivat, T., Trubert, B., Roy, D., Pierre, G.: MEC-ConPaaS: An experimental single-board based mobile edge cloud. In: Proc. IEEE Mobile Cloud Conference (2017). DOI 10.1109/MobileCloud.2017.17
18. Lucia, B., Balaji, V., Colin, A., Maeng, K., Ruppel, E.: Intermittent computing: Challenges and opportunities. In: Proc. SNAPL (2017). DOI 10.4230/LIPIcs.SNAPL.2017.8
19. Lufft: Compact weather sensors - WS501-UMB smart weather sensor. URL https://www.lufft.com/products/precipitation-sensors-287/ws100-radar-precipitation-sensor-smart-disdrometer-2361/. Last Accessed: 2025-07-01
20. Malossi, A.C.I., Schaffner, M., Molnos, A., Gammaitoni, L., Tagliavini, G., Emerson, A., Tomás, A., Nikolopoulos, D.S., Flamand, E., Wehn, N.: The transprecision computing paradigm: Concept, design, and applications. In: Proc. DATE (2018). DOI 10.23919/DATE.2018.8342176
21. NVIDIA Corporation: Jetson Nano | NVIDIA Developer. URL https://developer.nvidia.com/embedded/jetson-nano. Last Accessed: 2025-09-15
22. OTT Hydromet: OTT RLS – radar level sensor. URL https://www.ott.com/products/water-level-1/ott-rls-radar-level-sensor-861/. Last Accessed: 2025-07-02
23. Pagliari, A., Pierre, G.: TransScale: Combined-approach elasticity for stream processing in fog environments. In: Proc. MobileCloud (2023). DOI 10.1109/MobileCloud58788.2023.00009
24. Pearson, K.: X. contributions to the mathematical theory of evolution.—ii. skew variation in homogeneous material. Philosophical Transactions of the Royal Society of London. (A.) **186** (1895). DOI 10.1098/rsta.1895.0010
25. Prometheus Authors: Prometheus - monitoring system & time series database. URL https://prometheus.io/. Last Accessed: 2025-06-24
26. Red Hat Inc: An API/Schema registry – stores APIs and schemas. URL https://github.com/apicurio/apicurio-registry. Last Accessed: 2025-08-29
27. République: Flood sweeps away four bridges, damages two in upper Mustang (2025). https://myrepublica.nagariknetwork.com/news/flood-sweeps-away-four-bridges-damages-two-in-upper-mustang-90-49.html
28. Sarramia, D., Claude, A., Ogereau, F., Mezhoud, J., Mailhot, G.: CEBA: A data lake for data sharing and environmental monitoring. Sensors **22**(7) (2022)
29. Tashakkori, R., Hamza, A.S., Crawford, M.B.: Beemon: An IoT-based beehive monitoring system. Computers and Electronics in Agriculture **190** (2021). DOI 10.1016/j.compag.2021.106427
30. Turing Machines Inc.: Turing RK1 - compute module based on Rockchip RK3588. URL https://turingpi.com/product/turing-rk1. Last Accessed: 2025-09-15
31. Vaisala: Vaisala weather transmitter WXT520. URL https://www.vaisala.com/sites/default/files/documents/M210906EN-C.pdf. Last Accessed: 2025-07-01
32. Wikipedia: Gandaki river. https://en.wikipedia.org/wiki/Gandaki_River
33. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al.: The FAIR guiding principles for scientific data management and stewardship. Scientific data **3**(1) (2016)
34. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture **98** (2019). DOI 10.1016/j.sysarc.2019.02.009
35. Zacepins, A., Stalidzans, E., Meitalovs, J.: Application of information technologies in precision apiculture. In: Proc. ICPA, vol. 7 (2012)