Journal of Hydroinformatics

Journal of Hydroinformatics Vol 25 No 3, 851 doi: 10.2166/hydro.2023.147

A microservice architecture for leak localization in water distribution networks using hybrid AI

Ganjour Mazaev [a,*], Michael Weyns [a], Pieter Moens [a], Pieter Jan Haest [b], Filip Vancoillie[b], Guido Vaes [c], Joeri Debaenst[c], Aagje Waroux[d], Kris Marlein[d], Femke Ongenae [a] and Sofie Van Hoecke [a]

[a] IDLab, Ghent University – imec, 9052 Zwijnaarde, Belgium
[b] De Watergroep, 1000 Brussels, Belgium
[c] Hydroscan, 3010 Leuven, Belgium
[d] Itineris, 9000 Ghent, Belgium
*Corresponding author. E-mail: ganjour.mazaev@ugent.be

GM, 0000-0002-7494-1685; MW, 0000-0002-6157-5997; PM, 0000-0003-2035-8766; PJH, 0000-0002-5217-610X; GV, 0000-0002-6961-8736; FO, 0000-0003-2529-5477; SVH, 0000-0002-7865-6793

ABSTRACT

Up to 30% of all drinking water is wasted due to leaks in water distribution networks (WDNs). In times of drought and water shortage, wasting so much drinking water has a considerable environmental and financial cost. In this paper, we present a microservice architecture for leak localization in WDNs, where heterogeneous sources of data consisting of sensor measurements, Geographic Information System (GIS), and Customer Relationship Management (CRM) data are used to feed a leak monitoring solution which combines hybrid data-driven and model-based leak detection and localization methodologies. The solution is validated using in-field leak experiments in an operational WDN. The final leak probabilities are presented in a visualization dashboard. The search zone for most leaks is reduced to a few kilometers or less. For other leaks, the solution is able to indicate a larger search zone to reflect its higher leak prediction uncertainty.

Key words: hybrid AI, hydraulics, leak localization, machine learning, microservice, water distribution network

HIGHLIGHTS

- A microservice architecture for leak monitoring in WDNs is designed.
- Sensor measurements and GIS and CRM data are used for leak localization.
- A post-processing procedure is presented to combine leak location predictions of two hybrid model-based and data-driven methodologies.
- The architecture is evaluated using real leaks in an operational WDN.

ACRONYMS LIST

AI	Artificial Intelligence
API	Application Programming Interface
CSV	Comma-Separated Values
FTP	File Transfer Protocol
GPS	Global Positioning System
Ids	Identifiers
JSON	JavaScript Object Notation
MEMS	Microelectromechanical System
IoT	Internet of Things
i.a.	inter alia
REST	Representational State Transfer
UTC	Coordinated Universal Time
WSGI	Web Server Gateway Interface

1. INTRODUCTION

Water scarcity is currently affecting many metropolitan communities throughout the world. Over the next three decades, population growth, urbanization, and socioeconomic development are predicted to raise urban industrial and household water consumption by 50–80% worldwide (He *et al.* 2021). At the same time, up to 30% of all drinking water are wasted in the distribution of drinking water due to leaks, resulting from damaged or ageing pipes. Therefore, monitoring of water leakages within water distribution networks (WDNs) is of foremost importance for the sustainable development of urban cities (Xie *et al.* 2018). Water quality is also affected by leaks since they introduce low-pressure conditions into the WDN, which could result in contaminated water. An increased focus on real-time water quality monitoring, with the aim to monitor water more continuously, has been observed in recent years (Piazza *et al.* 2023).

Leaks in WDNs can be monitored through leak detection (LD) and leak localization (LL) techniques. LD methods determine whether or not a leak is present in a WDN. LL methods identify the location of a leak inside a WDN. LD is thus required before determining the exact location of the leak. A recent, systematic overview of LD and LL techniques can be found in Hu *et al.* (2021). Examples of LD methods include the work by Fan *et al.* (2021), who developed neural network models to distinguish WDN pressure data under leaking versus leak-free conditions, and the work of Tariq *et al.* (2022) where MEMS accelerometers are used for LD in pipe networks.

LL methods can be classified into model-based, data-driven, and mixed approaches (Quiñones-Grueiro *et al.* 2021). Model-based approaches use a hydraulic model of the WDN based on first principles to simulate pressure data. The approximate leakage location is determined by comparing in-field pressure data with their estimation obtained using the hydraulic model (Adedeji *et al.* 2017; Mohammed *et al.* 2021). Data-driven methods use historical data to construct a predictive LL model based on anomalies in pressures (Lučin *et al.* 2021). Mixed model-based/data-driven approaches use the combination of a calibrated hydraulic model with a classification method for LL. A classification method in machine learning (ML) is an algorithm that automatically learns to categorize data into classes. In this case, the training data for the classification method is generated by a hydraulic model of the WDN studied, where every leak location candidate is represented by a class. (Soldevila *et al.* 2016). Recent examples of hybrid LL approaches of this kind include the work by Mazaev *et al.* (2023), a methodology that combines hydraulic pressure simulations with an elastic-net logistic regression model, and the work by Weyns *et al.* (2022), using neural network autoencoders and leveraging Geographic Information System (GIS) data.

While a large body of academic literature on leak monitoring methods exists, which focuses on the development of new algorithmic methodologies, few works describe the software architectural implementation of these methods. For LD, a system using signal processing and ML is described by Singh *et al.* (2021). A network of acoustic and GPS sensors is used to continuously monitor the sound in the neighborhood of the pipe. Edge and cloud technologies are described, which allow the LD system to operate at low latency. In Ali *et al.* (2022), an IoT testbed architecture is designed, which presents LD problems to users. Commercial solutions for LD are also already available on the market, such as LeakRedux (Hydroscan 2022) and TaKaDu (Peleg 2022).

To the best of our knowledge, no architectural designs for LL have been described yet in literature, neither from academic nor commercial solutions. In this paper, we therefore present our design of a microservice architecture for LL in WDNs. The heterogeneous data sources consumed by this architecture consist of sensor data of pressures and flows, GIS data and Customer Relationship Management (CRM) data. One of the challenges for our design is bringing this data together in one unified architecture. This data is then used by two aforementioned, mixed LL methodologies. Our methodologies described in Mazaev *et al.* (2023) and Weyns *et al.* (2022) will be referenced as the *Physics-ML* and *GIS-ML* methods, respectively. A second challenge is bringing together the leak predictions of both methods.

Since each LL approach has its own strengths and weaknesses, we introduce a method in which their LL predictions are traded off against one another, resulting in a hybrid, combined prediction. We also show how these combined predictions can be further refined by using CRM data which provide useful information on leaks occurring in the network. We note that the resulting architecture is not limited to using these methodologies and can be used by other methodologies as well. Finally, a visualization dashboard is implemented to show the LL probabilities on a map of the WDN, next to relevant CRM data for the associated leak alarm. The LL performance of the architecture is tested by performing in-field leak experiments in an operational WDN by opening hydrants or closing valves.

2. METHODS

2.1. Data

The BK-Town District Metered Area (DMA), a WDN managed by De Watergroep in Belgium, was used to evaluate our LL methodology and architecture. The area of the DMA is 110 km^2, covering approximately 9,000 home connections. The total length of the network pipes is 217 km. The total water consumption in the DMA is 2,450 m^3/day on average. -Its network structure is shown in Figure 1. Hydrants and pipes are visualized as nodes and edges, respectively. All pipes are visualized using the same edge thickness, irrespective of their diameters, for visual clarity. Various sources of data were collected to monitor the DMA, with the end goal of localizing and visualizing leaks.

2.1.1. Sensor data

The type of sensor used falls under two categories: 'fixed' sensors and 'mobile' pressure sensors.

Flow, pressure, and water level sensors fall under the first category and are installed at fixed locations in the WDN. Their measurement frequency is 5 min. They are permanently installed at reservoirs and water production centers to monitor the WDN. Flow sensors are crucial at the boundaries of the DMA to obtain the water balance of the network. Level sensors in reservoirs are of added value since their measurements can be converted into pressures. As a result, no additional pressure

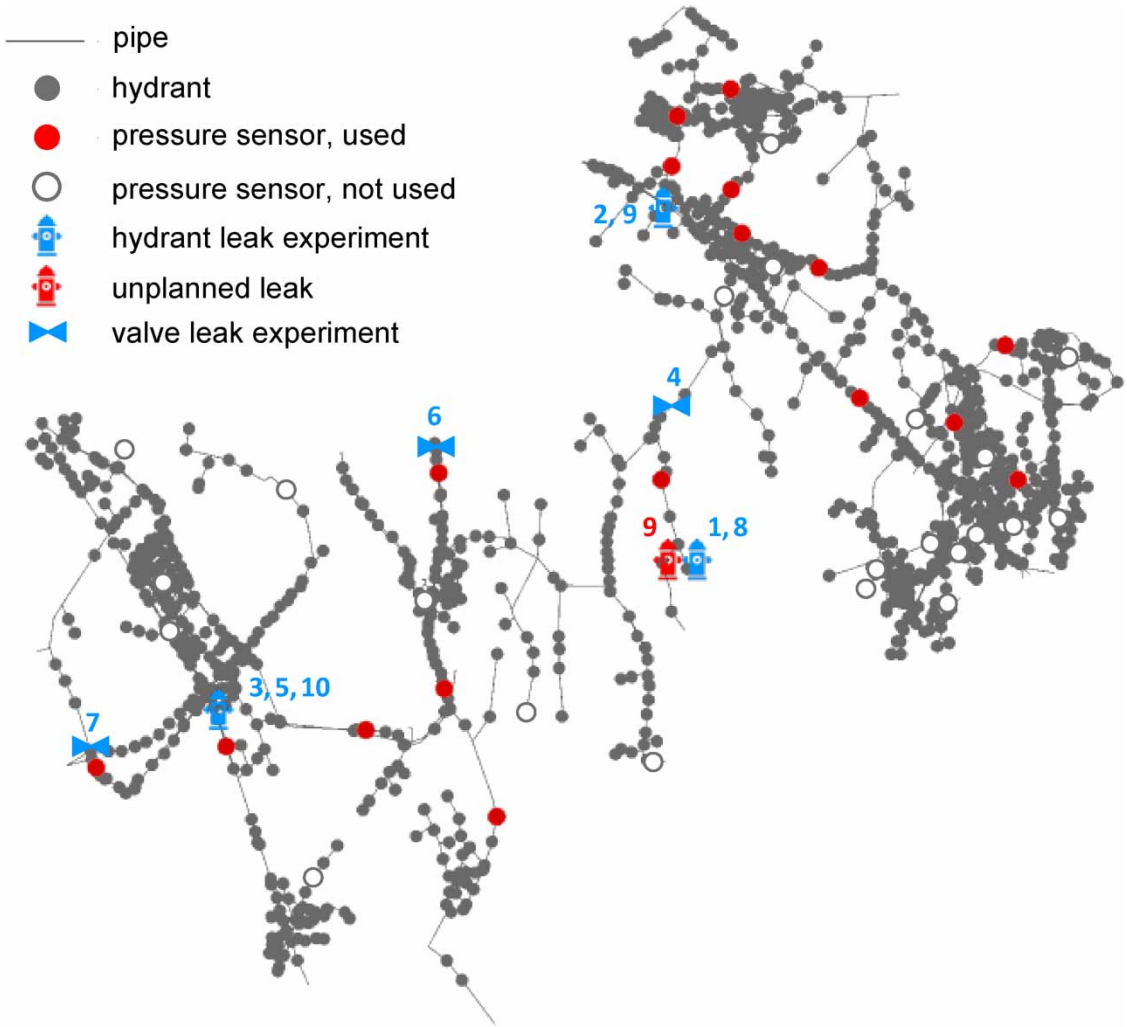

Figure 1 | Overview of the BK-Town WDN structure. The pressure sensor and leak experiment locations are indicated in the network. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.147.

sensors are needed at these locations. Within the context of this work, the main use of these sensors is for LD and hydraulic modeling, as discussed in Section 2.3.

Mobile pressure sensors were also installed in the network. They were placed specifically for providing pressure data to the LL architecture described in this work. These sensors are considered mobile compared to the fixed sensors as they can be placed flexibly at different hydrant locations in the network. These mobile pressure sensors are battery-powered, piezoresistive pressure transmitters which are placed in a street lid of an underground hydrant. The measurement frequency for these sensors is 1 min.

The measurement locations for the mobile pressure locations were chosen by domain experts in WDN leak management. It was ensured that the locations were sufficiently spread across the network, and that pressures were always measured upstream and downstream from pressure reducers. Some rules of thumb were also applied:

- Measurements were carried out at the ends of the network but on the smallest pipelines.
- The pressures at intersections of large pipes were measured.
- Measuring large pipelines with constant inlet pressures was avoided, due to the limited pressure response of a mid-size leak.
- Knowledge of specific types of pipelines (materials and age) that are more likely to cause greater load losses is taken into account.

An overview of the mobile pressure sensor locations chosen in the BK-Town network is shown in Figure 1. The mobile sensors were installed on September 17, 2021, and 2 weeks before leaks in the network were initiated by opening hydrants. Unfortunately, 22 of the 39 mobile pressure sensors encountered connectivity issues in this period, resulting in data packages not being transmitted. As a result, only 17 pressure sensors were used for LL, shown as red nodes in Figure 1. The remaining sensors are indicated in black.

2.1.2. Leak experiments

Leaks were induced in the WDN on purpose, with 10 leak experiments in total at 6 different locations. The location of the different leak experiments in the network is indicated in Figure 1. An overview of the timing and leak size of the experiments is given in Supplementary Table 1. At three locations, a closing valve is opened to a neighboring DMA at a lower hydraulic pressure. This action mimics a large leak of approximately 25 m^3/h. At the four other locations, a fire hydrant was opened with a flow meter to reach an outflow of 15 m^3/h. Examples of the hydrant leak experiment setup are shown in Supplementary Figure 1. Note that the locations with the hydrant leak experiments were selected in the vicinity of natural waterways or agricultural fields to prevent outflow to the sewer system.

For the hydrant location of experiments 3, 5, and 10 in Figure 1, it was already known that leaks at this location do not lead to detectable changes in any deployed pressure sensor in simulations carried out before the leak experiments. These experiments were included on purpose to be able to explicitly evaluate if the LL models are able to express a high LL uncertainty for this location.

In addition to the planned leak experiments, an unplanned leak occurred during experiment 9. This unplanned leak is indicated in Figure 1 as a red hydrant.

2.1.3. GIS data

A GIS allows us to model the location of various pipes and how they are connected within a larger network. Within the scope of this paper, a GIS was used that represents the BK-Town WDN. This GIS was provided by De Watergroep in the format of several CSV files, each corresponding to a specific type of element inside the water network. These elements are the following:

- customer point
- fixed head
- hydrant
- meter
- node
- non-return value
- pipe
- transfer node
- valve

Of the numerous properties associated with each element, we retain only those necessary for localization. For the various kinds of connectors (pipes, valves, non-return values, and meters), we keep the head and tail nodes associated with each of them (the corresponding node identifiers), as well as the length of the connector. For the various kinds of nodes (nodes, hydrants, customer points, fixed heads, and transfer nodes), we keep the node identifier, as well as the x and y coordinates. Using this information, we can easily set up a graph with nodes and edges connecting them (as shown in Figure 1), which is supplied with additional features such as edge lengths and node coordinates.

2.1.4. CRM data

De Watergroep manages its customer relationships (CRM) with a custom-built Enterprise Resources Planning (ERP) system: Neptunus. The ERP system is currently being transformed into the Microsoft Dynamics 365 environment that was already used for this research. It contains data about the client, client interactions, and customer-centric assets such as the indoor service line, water meter, and outdoor supply line up to the tapping saddle. Following General Data Protection Regulation (GDPR) regulations, client data were not disclosed in this research, and use was made of dummy data and client interactions following the data model of the CRM module in Neptunus. Client interactions in this research are obtained through phone calls, chatbot, and/or social media. The latter two are set up as the first access point, where the client is subsequently put through to a customer officer of the water utility.

A customer officer logs client interactions with the creation of a '*case*'. This *case* ties the interaction to the customer information and the subsequent actions by the water utility. Currently, two types of *cases* are discerned: '*contact case*' and '*normal case*'. The difference between the two case types is the physical action that is taken by the water utility in a '*normal case*' to follow up on the client interaction. If a physical action is required, a '*service order*' is created. Different types of service orders exist, e.g. '*evaluation*', '*maintenance*', '*leak*' and/or '*road repair*'. Following a successful action, the client is informed and the *case* is closed.

The interactions with the clients provide valuable information for LD of current leaks and WDN history. As described above, this information is heterogeneous in nature and origin: complaint of water pressure versus visual observation of water on the street and information from skilled employees versus information from discomforted customers. Therefore, this information was made available for the LL algorithms with a weight factor, which was derived from expert judgement (see Section 2.3.5).

2.2. Data exchanges between data sources and shared data storage

For real-time or near-real-time applications, a uniform and well-defined data infrastructure is needed, where data are stored and transmitted in an organized manner. In data-driven LL, ample data in the form of time-series are considered. The data infrastructure needs to allow for adequate scalability and throughput for this type of unstructured data (O'Leary 2014).

An overview of the data architecture is schematized in Figure 2, visualizing how data are stored and communicated. The data sources consist of GIS, CRM, and sensor data, which were discussed in Section 2. The data are stored in a shared data storage (SDD) using Azure Blob Storage. Leak monitoring APIs connect to the SDD to retrieve the data needed for LD and LL and to store their results. The functionality of the LeakRedux LD module is described in more detail in Section 2.3.1. The hybrid ML LL APIs, triggered by the LD module, are also described in more detail in Section 2.3.2.–2.3.5.

Finally, the visualization dashboard is provided in Dynamics 365 Customer service. It retrieves the LL results to visualize predicted leak locations toward experts responsible for finding the leak. Leak location probabilities associated with a leak alarm are visualized on a map of the DMA, next to relevant service orders and cases relevant to that particular leak alarm. More details are given in Section 2.3.6.

2.2.1. Data sources

GIS-related data were not continuously updated in this data architecture. During the timespan of setting up the architecture and using it for monitoring leak experiments, changes to the WDN GIS structure were limited to repairs required for maintenance. Therefore, the GIS was fixed to the version with which the hydraulic model was developed. The GIS data were manually exported once to the SDD. In future designs, the architecture may be extended to automate this step.

The data from the fixed sensors are included in the operational processes of De Watergroep. These data were diverted to the SDD as soon as they reached the data warehouse (in Azure Data Lake Analytics) through Azure Data Factory. On the other hand, the data from the mobile sensors that were obtained specifically for evaluating this data architecture are not

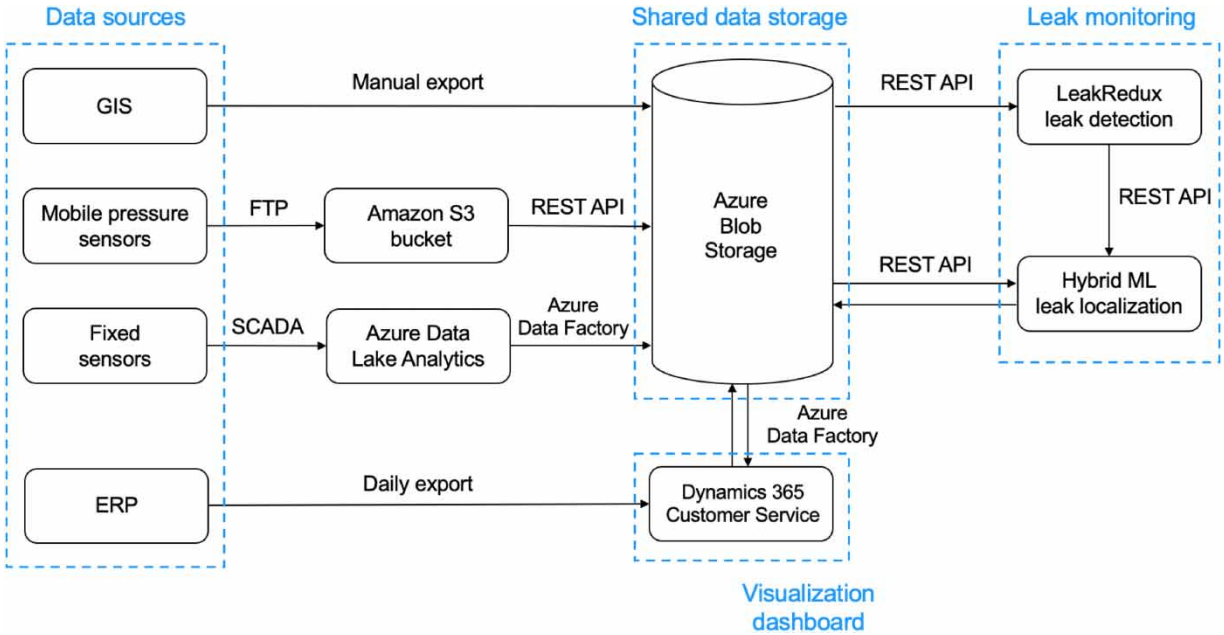

Figure 2 | Overview of the data architecture.

yet included in operational processes of De Watergroep. As such, these data were sent to an Amazon S3 bucket using FTP and then forwarded to the SDD.

CRM data are exported daily from Neptunus, the CRM system of De Watergroep, to the Dynamics 365 Customer Service environment for visualization in the dashboard. Azure Data Factory allows for selecting which fields of the entities in the customer service environment are exported to be forwarded to the SDD. As such, GDPR sensitive fields are excluded from the data exchange. The data export is limited to service orders and cases, which is to be collected by the hybrid ML LL API.

2.2.2. Shared data storage

The SDD was defined outside of the operational framework of De Watergroep to circumvent technical restrictions for data access/processing by external partners. Each research partner had a directory on the SDD with read/write privileges. Every partner had read-only access to the other partner's directories.

The SDD is a file-based server in Azure Blob Storage. Thus, data are shared in the form of unstructured data files. Sensor data of both mobile and fixed sensors are stored in the form of CSV files. Parameters and settings which are relevant for inference after training of the LL models are stored in the form of serialized Python pickle files.

2.3. Leak monitoring architecture

An overview of the leak monitoring architecture is given in Figure 3. LD is performed by the LeakRedux software. The LL functionality is implemented in four microservices: Ingress, Physics-ML, GIS-ML, and Egress. All input data are retrieved from the SDD, as described in Section 2.2.

The LL microservices are implemented using Kubernetes, a container-orchestration system. Every service is implemented in Python and served as a REST API using Flask, a micro web framework. The Flask applications are deployed within a Python WSGI server (Gunicorn) which allows concurrent request handling.

Every API is containerized in a Docker container, which is then orchestrated in the Kubernetes cluster. Gitlab Continuous Integration and Delivery/Deployment was used to automate building, testing, and deployment. Physics-ML and GIS-ML also make use of Celery (task queue software) and RabbitMQ (message broker), so that its jobs can be processed asynchronously, which means that the execution thread of a client is not stalled after making a request. Due to the asynchronous execution of long-running jobs using Celery, additional workers can be spawned to execute the tasks in the queue in parallel, making the architecture scalable. The components are discussed in more detail in the next subsections.

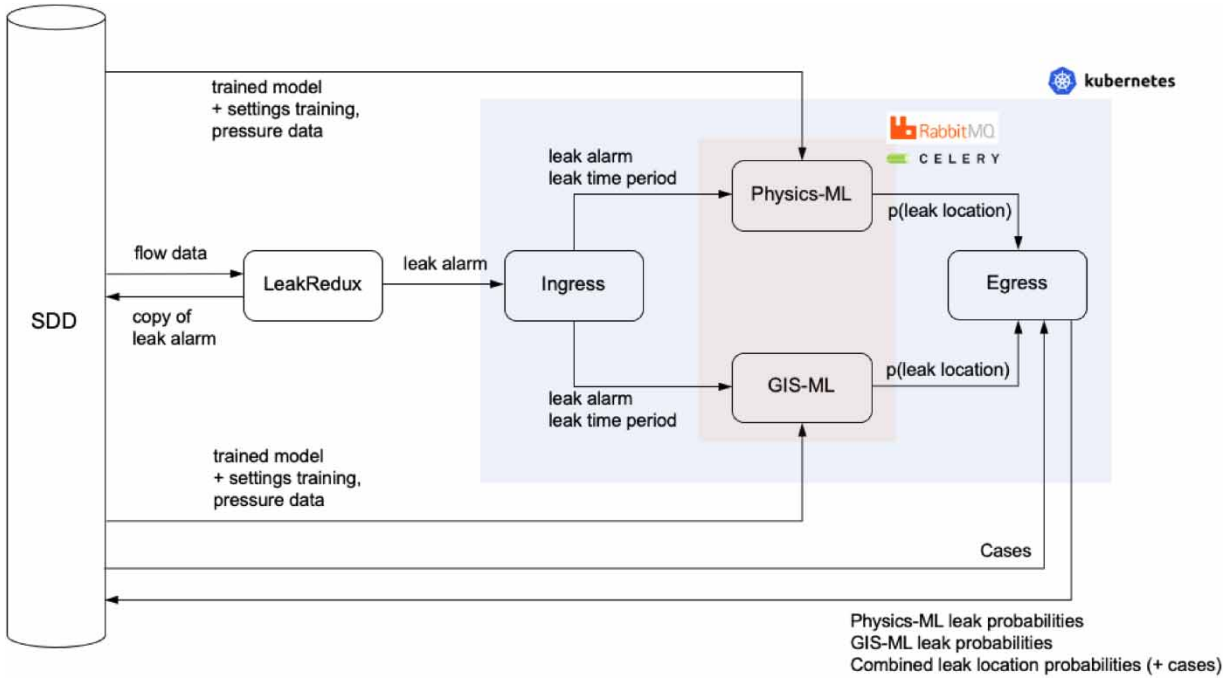

Figure 3 | Overview of the leak monitoring architecture.

An important advantage of deploying the models as modular Docker containers within a Kubernetes cluster is that models can be easily modified, added, or replaced, highlighting the added value of the proposed architecture for future LL designs.

2.3.1. Leak detection

LeakRedux is a software solution by Hydroscan used to detect i.a. the occurrence of medium and large bursts and estimate the live background leakage in a DMA (Hydroscan 2022). Flow measurements in a DMA are used to construct an online mass balance to follow up water use in the function of time. Sudden and continuous increases or drops in the mass balance yield indications for leakages, changes in network behavior, or leak repairs. Information about historical leaks is used as input to construct an initial leak-free reference of the DMA.

Based on new data that are available on the SDD, LeakRedux calculates potential leak alarms for every time-step of the fixed sensors, having a measurement frequency of 5 min. For running alarms, parameters are updated. When a leak alarm is generated by LeakRedux, it is sent in real-time to the ingress entry point of the LL API in JSON format. The LL model is thus triggered by an alarm containing all necessary details about the leak, such as the leak size, and when the leak has started. For logging purposes, a copy of every alarm is written to the SDD.

2.3.2. Ingress

The Ingress API listens to LeakRedux, which is communicated as a JSON file. Ingress forwards the leak alarm and the leak period to the Physics-ML and GIS-ML services, hence triggering the LL functionality.

2.3.3. Physics-ML

The **Physics**-ML API uses a hybrid LL approach, which combines both model-based and data-driven modeling. Pressure heads of leak scenarios are simulated using a hydraulic model of the WDN and then used to train a ML-based LL model. This methodology is described in detail in Mazaev *et al.* (2023), where it is evaluated on real leak experiments in two operational DMAs. By calculating the shortest path length between the actual leak location and the highest predicted probability for every leak experiment, the model's LL performance was assessed numerically. These shortest path lengths ranged from 0.18 to 4.96 km, and from 0.94 to 17.56 km, for the 1st and 2nd DMA, respectively.

A key component of the Physics-ML methodology is that discrepancies between simulated and measured pressures are corrected for using historical pressure measurements, which is called the Time-Windowed Head Bias Correction (TWHBC). The methodology is described here in summarized form by using its mathematical formulation.

We assume N potential leak locations, with each location indicated by i, from 1 to N. We assume M pressure sensors in the WDN, with each sensor indicated by j, from 1 to M. Days are indexed by k, where $k = K$ denotes the day on which a real leak occurs and the TWHBC is calculated. In the following formulas, we implicitly assume a specific time-window (e.g., 9:15–9:30) in order to prevent overburdening the notation. The time-step frequency for all pressure head time-series (both for leaky and leak-free scenarios) corresponds to the measurement frequency of the fixed sensors described in Section 2.1.1., which is 5 min. The pressure head time-series within the time-window are condensed to one value by calculating their mean over all time-steps within the time-window. For example, the four time-steps in window 9:15–9:30 are reduced to their mean value.

The head values and residuals for sensor j and day k within a time-window can then be defined as follows:

- $h^m_{j,k}$ = the measured head.
- $h^0_{j,k}$ = the leak-free simulated head.
- $h_{i,j,k}$ = the simulated head for leak location i.
- $r^0_{j,k} = h^0_{j,k} - h^m_{j,k}$ = a head residual corresponding to the leak-free simulation.

The head residuals corresponding to the leak-free simulation, averaged over days $k = 1$ to $k = K - 1$, provide the means of the TWHBC:

$$\bar{r}^0_j = \frac{1}{K-1}\sum_{k=1}^{K-1} r^0_{j,k} \tag{1}$$

with its associated standard deviations:

$$\sigma_j = \sqrt{\frac{1}{K-2}\sum_{k=1}^{K-1}\left(r^0_{j,k} - \bar{r}^0_j\right)^2} \tag{2}$$

The following values are calculated on day K:

$$\mu_{i,j} = h^0_{j,K} - h_{i,j,K} \tag{3}$$

The calculated values in Equations (2) and (3) are used to construct the following multivariate Gaussians with a diagonal covariance matrix for each leak candidate i:

$$\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}) \text{ with } \boldsymbol{\mu}_i = \begin{bmatrix} \mu_{i,1} \\ \cdots \\ \mu_{i,M} \end{bmatrix}; \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \sigma_M^2 \end{bmatrix} \tag{4}$$

The features for each leak candidate i are then generated by randomly sampling its associated Gaussian distribution. Each distribution is sampled 40 times to generate a training data set. The features of this training data set are then standardized to their means equal to 0 and variance equal to 1. An elastic-net logistic regression is trained on this data set, where each leak candidate i represents a class of the classification model.

The head residuals corresponding to the pressure sensor measurements at day K are calculated as follows:

$$x_j = h^0_{j,K} - \bar{r}^0_j - h^m_{j,K} \tag{5}$$

After standardization, these head residuals are fed to the trained elastic-net logistic regression model, which predicts a leak probability for each leak candidate i.

For the purpose of the leak monitoring architecture, the Physics-ML model was first trained offline. Leaks of 15 m^3/h were simulated in every hydrant of BK-Town. This totals $\pm$1,000 simulations since there are $\pm$1,000 hydrants in the WDN. The leak-free scenario is also simulated. Using this simulated data, the Elastic-Net logistic regression LL model is trained. This model is then able to assign a leak probability for every hydrant based on incoming pressure measurements in real time, which is converted through Equation (5).

As a post-processing step, leak probabilities at the hydrant level are converted to probabilities at the pipe level to allow for a clear visualization of the leak predictions. This conversion was done for every pipe by assigning the probability of the nearest hydrant to the pipe and then normalizing all pipe probabilities by their total.

Once the Physics-ML API is triggered by the Ingress API, the trained Physics-ML models and mobile pressure sensor data are imported from the SDD. During training, a leak classification model was trained for every 15-min interval of the day. Based on the start and end times of a leak as forwarded by the Ingress API, one of the models is used for the leak prediction. This is illustrated in Figure 4. In this example, the leak starts at 9:10 and ends at 9:35. The model constructed for the interval 9:15–9:30 is used since it falls within the leak interval. In general, the Physics-ML API utilizes the most recent model that falls within the LD interval. The motivation for using time intervals with a separate trained Physics-ML model per interval is that the TWHBC captures uncertainties in the simulated pressure heads which are the most relevant for a particular time-window of the day. Thus, our approach ensures that appropriate uncertainties are captured for a proposed LD interval.

The Physics-ML API also loads some settings from the SDD, which were set during the training phase:

- the pressure loggers that were used as ML features,
- the time intervals for which the models were trained,
- the time-windowed pressure head bias correction per pressure sensor,
- the conversion needed to convert hydrant probabilities to pipe probabilities.

The Physics-ML API finally sends its leak probabilities to the Egress API together with the 'AlertCode' of the leak alarm. This enables the Egress API to link the LL results with the associated leak alarm.

2.3.4. GIS-ML

The GIS-ML API uses a mostly data-driven approach, with simulations being used only to generate replacements for any missing values in the input data. An underlying ML model is used based on the architecture of autoencoders. This model is trained in an offline fashion on a stretch of data presumed to reflect leakless network conditions. By doing this, the model is able to reconstruct leakless data with high accuracy while exhibiting significant reconstruction errors on aberrant data sequences. For more details on how this methodology works, refer to Weyns *et al.* (2022).

During deployment, GIS-ML will first receive a start date, an end date, an alert code, and a DMA code from the Ingress API, similar to the Physics-ML model (see Figure 4). Before doing anything else, it will fetch the most recent sensor data from the SDD and clean it, so that it pertains only to the sensors for which the model has been trained for. After the cleaning step, the data are preprocessed further to deal with any missing values. This involves fetching the most recent hydraulic model inputs, generating a new file describing the network's current boundary conditions, running a leakless simulation using this file, and then filling in the missing values with the corresponding simulation values. It also involves shifting the

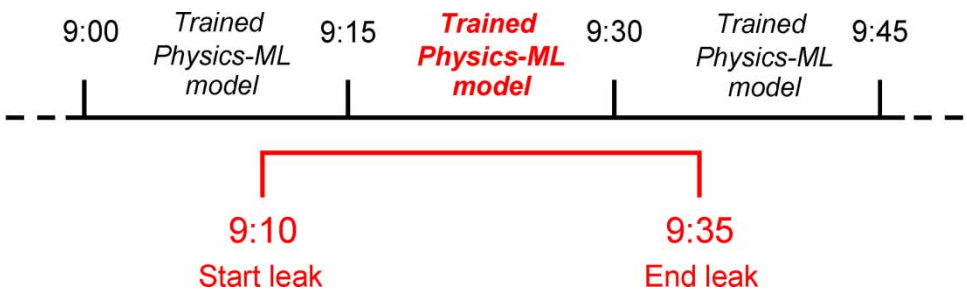

Figure 4 | Illustration of the intervals for which Physics-ML models are constructed. In this case, the model for the time interval from 9:15 to 9:30 is used based on the LD interval.

timestamps of the sensor data to UTC (the time zone used by the Ingress alerts) and converting the pressures (in bar) to head-loss values (in mH_2O).

After these preprocessing steps, we select a period of data corresponding to the start and end dates provided by the Ingress API. Because the model was trained on time-series with 5-min intervals between time stamps, we want to make sure we select our data in a way that corresponds to this sampling frequency. As such, we select a slice of data by decrementing the start and end dates until they each match timestamps that are multiples of 5 min. During this step, we also take into account the window size used by the underlying ML architecture, as this corresponds to the minimal number of samples, a data frame must have in order to be suitable for prediction. We do this by decrementing the start date until the number of samples that are 5 min apart is larger than the window size.

Given this slice of data, we apply the autoencoder-based ML model to generate reconstruction errors for each of the input sensors. Next, the errors are distributed throughout the network. To augment this process, we make use of the DMA code provided to us by skipping over reconstruction errors belonging to sensors associated with the wrong DMA code. This makes it so that these sensors have no impact at all on the final probability distribution. After the results have been generated (a max-normalized probability distribution over all the pipes in the network), we send them over to the Egress API together with the name of the API and the alert code. An example of this could be 'gis_ml_prod' as API name and the original alert code 63750.

2.3.5. Egress

So far, we have discussed two primary alternatives to the LL problem. One of the approaches makes use of a more complex hybrid ML methodology reliant on hydraulics, while the other approach applies ML more directly, thereby reducing any knowledge of hydraulics to what can be learned from the telemetric data itself. It is important to note that the overall architecture is not limited to the implementation of these two approaches and can easily be adapted to other LD and LL strategies. Within the current setup, both approaches have their own strengths and weaknesses. Without any predetermined notion of which approach might be the most successful in any given situation, it becomes desirable to find a method with which they can be traded off against one another.

The Egress API is responsible for the melding together of the two individual LL approaches. As such, its functionality consists of a two-step post-processing procedure. First, we will discuss how the Egress hybridizes the individual model outputs. Then, we will also explain how it is used to incorporate additional, external information (in this case: customer information supplied by Itineris) to further enhance localization.

The Egress API is set up using precomputed partitions and coordinates for all nodes in the World Geodetic System (WGS84) coordinate system. It makes use of two separate worker threads each waiting for model predictions which should be composed in the following json format:

```
'Predictions': { 'predictions': {},
                 'source': '',
                 'alert_code': ''}
```

The prediction field should contain dictionary mapping pipeline IDs to probability scores. The source field should contain a string with the name of the sender API (either *gis_ml* or *physics_ml*). The alert_code field should also contain a string, mirroring the code provided to the Ingress for the current leak event.

The Egress API uses a conditional check to see how it should proceed once it has received predictions from either model. If either no task has been registered so far, or the previously registered task has been completed, or no predictions associated with the current alert code can be found, then the Egress starts a new task with a countdown timer of 10 min, registers the task, and stores the predictions it has received under the corresponding alert code. If, on the other hand, it receives predictions, but none of these conditions holds, then it will revoke the currently registered task, deregister this task, merge the newly received data with results previously stored under the current alert code, and start a new task with the combined results. It will then remove any association with this alert code. A representation of the flow of this procedure can be found in Supplementary Figure 2.

When everything goes well, every alert code should eventually trigger the second branch of the conditional. The task that is initiated will combine (hybridize) the results received from the model APIs. This is done by going over all the pipes in the

Table 1 | Overview of scores assigned per case

Case source type \ Case description	Fork/damage/consumption management	Pressure	Water in the street	No water	Misc.	Intake/ distribution
Administration/head of administration/ employee/Service task	0.8	0.35	0.9	0.8	0.7	1.0
Telecommunication	0.7	0.25	0.8	0.7	0.6	1.0
Chatbot	0.5	0.2	0.6	0.5	0.4	1.0
Social media	0.6	0.2	0.7	0.6	0.5	1.0
Misc.	0.6	0.2	0.7	0.6	0.5	1.0
Empty	0.6	0.2	0.7	0.6	0.5	1.0

network and averaging, on a per-pipe basis, the probability scores received from the Physics-ML and GIS-ML APIs, respectively. After the results have been combined, the API will fetch customer cases from the SDD, get the closest node for each relevant case (based on the latitude and longitude associated with each case), and apply customer information to the previously combined prediction results. In what follows, we will explain how this happens in more detail.

Customer cases are each associated with a number of information fields (see Section 2.1.4). Of these fields, only a few are relevant to the Egress model. These are: *Case description*, *Case category*, *Creation date*, *Phase*, *Longitude*, and *Latitude*. First, the creation date is used to check whether the case is still relevant or not. What constitutes relevancy in terms of temporality can be determined with reference to specific situations. After we have been able to determine whether the case is recent enough, we need to assign a score to it. This score is used to integrate the customer information with the computed leak probabilities in the Egress API. To assign a score to a given case, we make use of the score matrix in Table 1, which was set up by the experts at Itineris based on historical, in-field experience. In this table, the magnitude of the scores reflects a combination of the level of trust we can place in the source and the level of gravity reflected by the case description.

In this matrix, the columns express the *Case description*, while the rows express the *Case source type* – so roughly, the case description and the source via which the case was reported. Given that the necessary fields were actually supplied with the case, it becomes straightforward to assign a score to the case in question. All that remains now is to convert this score to actual leak probabilities associated with pipelines inside the network. To do this, we presuppose a *supplementary* interpretation of the information supplied via cases. This amounts to the following procedure:

1. Based on the latitude and longitude supplied with the case information, determine the closest node in the water network topology.
   a. The coordinates for the network nodes are given in the projected coordinate system for Belgium (the Lambert 72 coordinate system or EPSG:31370), while the latitude and longitude of the cases are given in the WGS84 format. A conversion is required.
   b. After conversion, compute the pairwise distance between the case's latitude–longitude pair and the (converted) latitude–longitude pair of every node in the network. Find the shortest distance.
2. Percolate the case score throughout the network.

   a. After distribution, each node in the network will have a portion of the original score associated with it (depending on the distance of the node from the case node).
   b. Convert the node scores to pipe scores by assigning to each pipe the average of the corresponding head and tail node scores.
   c. Compute the final score per edge as $score = \min(original\ score + case\ score, 1.0)$. The final score is therefore straightforwardly computed as the sum of the original pipe score and the case score as computed earlier for the given pipeline. Assign the final score to the relevant pipeline.

Finally, everything is written to the SDD in two files. One file will contain a single column of only the most recent (i.e., encountered since the previous write operation) alert codes, while the other is appended to using seven different columns: *guid*, *alert code*, *pipe id*, *gis score*, *physics score*, *hybrid score*, *customer score*, where the first column will contain a different

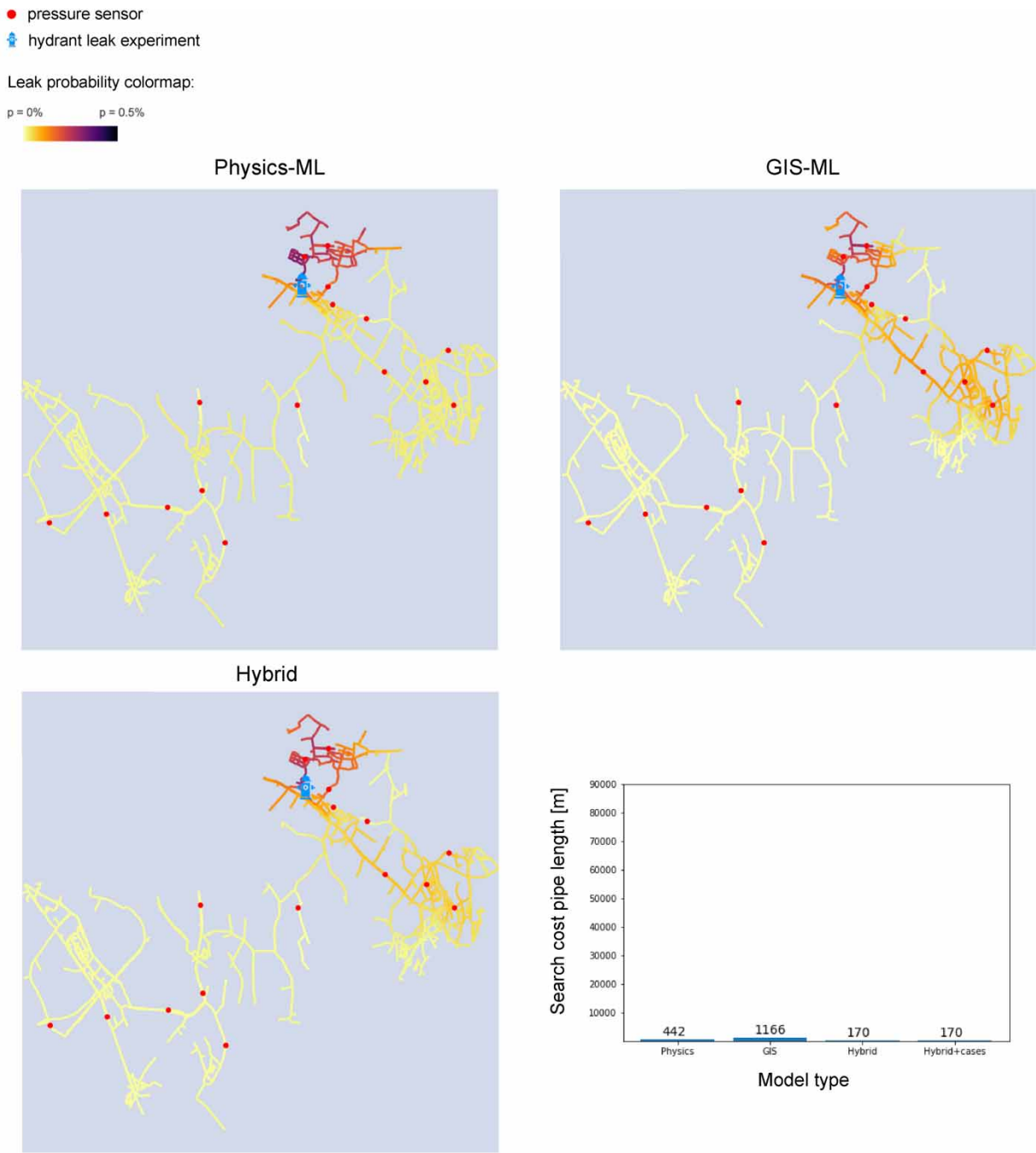

Figure 5 | LL results for experiment 2.

GUID for each row in the file. To be clear, the *gis score, physics score, hybrid score,* and *customer score* columns contain, respectively, the probabilities received by the Egress API from the GIS-ML API, those received from the Physics-ML API, the hybridized results computed by averaging the previous two columns on a per-pipe basis, and finally, the hybridized results augmented with supplementary customer information.

2.3.6. Visualization

A dashboard was created in the Customer Service module of Microsoft Dynamics 365. It visualizes the hybrid model output, next to customer alerts and service orders. In an operational context, it can be used by water distribution

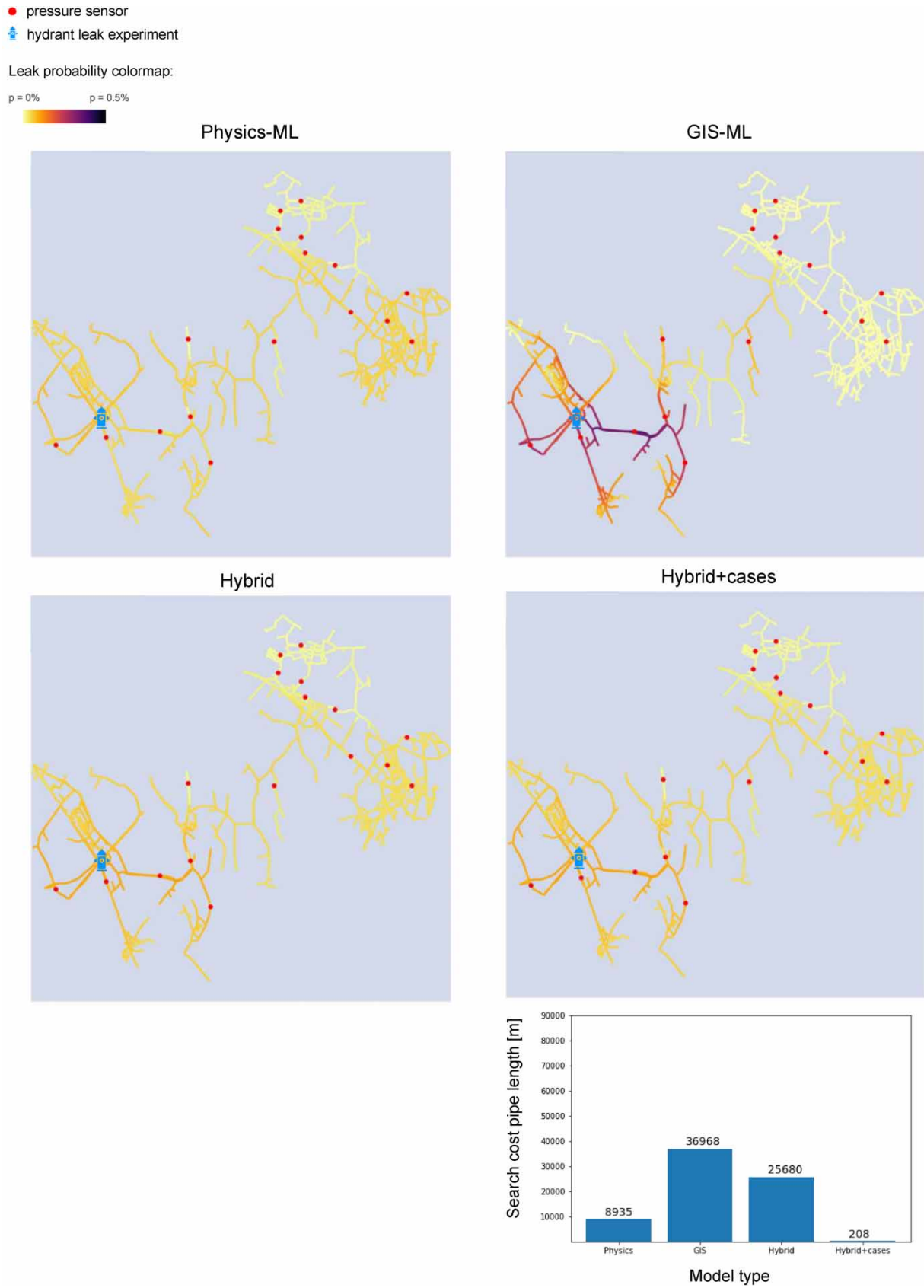

Figure 6 | LL results for experiment 5.

asset managers and customer service agents to have a visual overview of leak monitoring information. The dashboard uses data from Neptunus (the ERP system of De Watergroep), the Customer service environment itself, GIS data, and hybrid model data.

A selection can be made between different outputs: GIS-ML model, Physics-ML model, hybrid model, and hybrid + cases model. These represent the various models discussed in Section 2.3. The cases and service orders that are created in the demo environment are also shown on the map. An example is shown in Section 3.2.

3. RESULTS AND DISCUSSION

All LL results for the different methodologies are given in Supplementary Figures 3–12 in the Supplementary Material of this paper. Results for leak experiments 2 and 7 have been included in this paper in Figures 5 and 6. The location of the real leak is shown for every experiment, together with the leak probability assigned per pipe. These probabilities are encoded on a color scale, ranging from pale yellow (lowest probability) to dark red (highest probability), thus showing how each model distributes the leak probabilities. The search cost per methodology is also shown for every leak experiment, which is expressed in terms of pipe length (m) to be investigated in detail (in the order suggested by the leak probabilities outputted by the respective models) on the terrain to finally pinpoint the leak.

In general, good LL results are achieved, where the real leak locations are found in high-probability regions. For experiment 2, Physics-ML and GIS-ML both isolate the region of the WDN where the leak was simulated. The result is a hybrid prediction with a search cost of only 170 m (out of a total of 210 km). A similar observation can be made for Supplementary Figures 3, 6, 8, 9, 10, and 11. The hybrid model behaves as expected, offering a balance between the results of the two individual models.

The most problematic leak location is the hydrant for leak experiments 3, 5, and 10. For these experiments, we observe that the leak regions predicted by the hybrid model still comprise a large area of the WDN, which is up to a maximum of 27 km. As mentioned in Section 2.1.2, this is to be expected from the LL models, since simulations carried out before the leak experiments indicated that the sensitivity of the pressure sensor measurements is very low to leaks at this hydrant location. The Physics-ML model in particular outputs very uncertain predictions for these experiments, which is carried over to the hybrid predictions.

For leak experiment 5 (Figure 6), a customer case with source 'social media' and description 'leaking water' (score 0.5) was introduced in close proximity to the leak. This provides a very significant boost in performance, which is evidenced by a sharp drop in the search cost from 25,680 to 208 m. Another customer case, this time with source 'social media' and description 'water in the street' (score 0.7) was introduced for experiment 7 (Supplementary Figure 9). Here, the performance boost is less dramatic since a search cost of 1,842 m is already obtained by the hybrid model without cases.

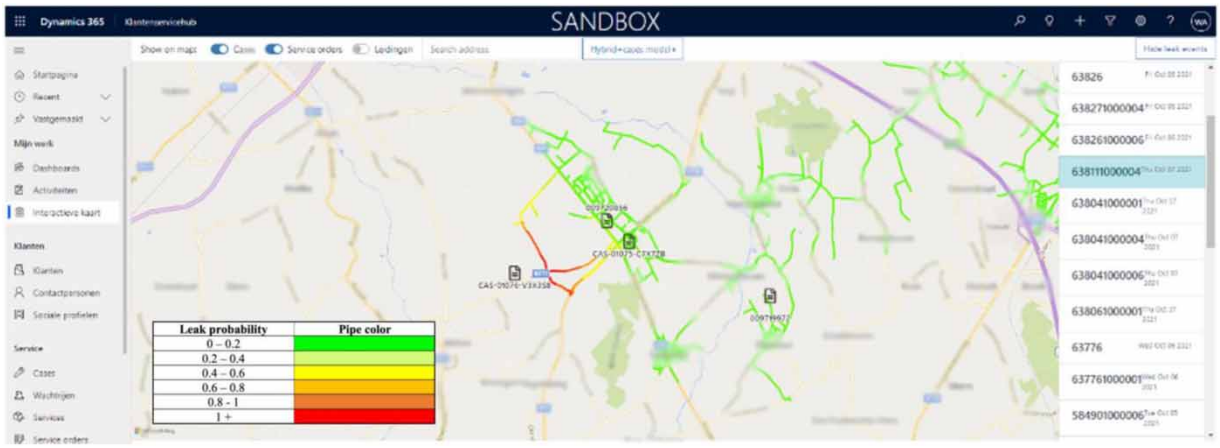

Figure 7 | Dashboard visualization for experiment 7. Geographical names are blurred. Please refer to the online version of this paper to see this figure in colour: http://dx.doi.org/10.2166/hydro.2023.147.

3.1. Multiple simultaneous leaks

A significant drawback of the proposed leak monitoring architecture became apparent for experiment 9. As mentioned in Section 2.1.2, an unplanned leak occurred next to the planned leak experiment, resulting in two simultaneous leaks. Physics-ML makes use of logistic regression, a discriminative classification model. The model was trained for singular leak locations only. The GIS-ML model makes use of the magnitude of reconstruction errors to figure out which locations most likely correspond to a leak. This means that, if the magnitude of a given leak signature dominates that of another simultaneous leak, only one leak will be clearly displayed. As a result, both models assign the highest leak probabilities to the unplanned leak. This effect is the most pronounced for Physics-ML, with a high search cost of 17,164 m.

Chances are small that multiple simultaneous leaks occur in one DMA. But often hydraulic pressure changes occur due to manipulations in the network. The hybrid model could then falsely estimate the manipulation as a leak, resulting in an increased lag in the response time for the actual leak. We conclude that the LL methodologies should be adapted to account for the occurrence of multiple leaks if we wish them to be able to localize multiple leaks in parallel.

3.2. Dashboard visualization

Figure 7 shows the results of leak experiment 7 in the dashboard. The dashboard displays the output of the hybrid model, as well as the ongoing cases representing the customer alerts. Geographical names have been blurred. The dashboard provides a clear view of where a technician should look first to pinpoint the leak, which are the pipes indicated in red, according to the supplied color code.

4. CONCLUSIONS

The proposed leak monitoring setup in this paper uses a combination of expert knowledge in the form of GIS and hydraulic modeling, together with signals from fixed and mobile sensors and customer interaction. We have demonstrated the effectiveness of this setup in an operational setting by showing effective LL predictions returned by the hybrid model. Despite the large amount of mobile pressure sensors that lost connectivity in our operational setup, effective LL results were still achieved using our methodology. The modular design of the LL APIs allows for flexible adaptations of the proposed architecture since the LL APIs can easily be modified, added, or replaced, which might be of interest for future leak monitoring designs.

Without a clear way to determine or even estimate which LL model might perform best in a given scenario, we must be able to mitigate the bad predictions while also retaining good ones. For this purpose, we devised a hybrid model that is able to combine the predictions of both singular models, while also taking into account the spread of the probabilities after the combination. Following an evaluation across the same leak scenarios, we were able to confirm the reliability of the hybrid model in situations where either one of the models clearly outperformed the other, thus achieving a nice performative balance. Continuing along this same vein, we also showed how customer information could be introduced to further enhance the results of the hybrid model. The final dashboard visualization then provides a clear view of the most probable leak locations.

However, the hybrid model cannot cope with multiple simultaneous leaks. The Physics-ML model is trained to detect one leak, and the GIS-ML model uses the magnitude of the reconstruction errors to define the probability. Future work on these methodologies could focus on adapting the methods to detect multiple concurrent leaks. Additionally, we might experiment with the integration of expert knowledge relating model performance to specific leak types, in order to perform a weighted averaging scheme which prefers the model most likely to perform well on a given leak.

ACKNOWLEDGEMENTS

The work of G. Mazaev and M. Weyns was supported by Research Foundation – Flanders (FWO) under strategic basis research doctoral grants (1S88020N, 1SD8821N). This work was supported through the SmartWaterGrid project, an imec.icon research project funded by imec and Agentschap Innoveren & Ondernemen.

DATA AVAILABILITY STATEMENT

Data cannot be made publicly available; readers should contact the corresponding author for details.

CONFLICT OF INTEREST

The authors declare there is no conflict.

REFERENCES

Adedeji, K. B., Hamam, Y., Abe, B. T. & Abu-Mahfouz, A. M. 2017 Towards achieving a reliable leakage detection and localization algorithm for application in water piping networks: an overview. *IEEE Access* **5**, 20272–20285.

Ali, A. S., Abdelmoez, M. N., Heshmat, M. & Ibrahim, K. 2022 A solution for water management and leakage detection problems using IoTs based approach. *Internet Things* **18**, 100504.

Fan, X., Zhang, X. & Yu, X. 2021 Machine learning model and strategy for fast and accurate detection of leaks in water supply network. *J. Infrastruct. Preserv. Resilience* **2**, 1–21.

He, C., Liu, Z., Wu, J., Pan, X., Fang, Z., Li and, J. & Bryan, B. 2021 Future global urban water scarcity and potential solutions. *Nat. Commun.* **12**, 4667.

Hu, Z., Chen, B., Chen, W., Tan, D. & Shen, D. 2021 Review of model-based and data-driven approaches for leak detection and location in water distribution systems. *Water Supply* **21** (7), 3282–3306.

Hydroscan 2022 LeakRedux. Available from: https://www.hydroscan.eu/en/leak-detection-leakredux-water-supply-network/ (accessed 5 September 2022).

Lučin, I., Lučin, B., Čarija, Z. & Sikirica, A. 2021 Data-driven leak localization in urban water distribution networks using big data for random forest classifier. *Mathematics* **9** (6), 672.

Mazaev, G., Weyns, M., Vancoillie, F., Vaes, G., Ongenae, F. & Van Hoecke, S. 2023 Probabilistic leak localization in water distribution networks using a hybrid data-driven and model-based approach. *Water Supply* **23** (1), 162–178.

Mohammed, E. G., Zeleke, E. B. & Abebe, S. L. 2021 Water leakage detection and localization using hydraulic modeling and classification. *J. Hydroinf.* **23** (4), 782–794.

O'Leary, D. E. 2014 Embedding AI and crowdsourcing in the Big Data Lake. *IEEE Intell. Syst.* **29** (5), 70–73.

Peleg, A. 2022 DMA monitoring. Available from: https://www.takadu.com/solution (accessed 14 July 2022).

Piazza, S., Sambito, M. & Freni, G. 2023 Analysis of optimal sensor placement in looped water distribution networks using different water quality models. *Water* **15** (3), 559.

Quiñones-Grueiro, M., Ares-Milián, M. J., Sánchez-Rivero, M., Silva Neto, A. J. & Llanes-Santiago, O. 2021 Robust leak localization in water distribution networks using computational intelligence. *Neurocomputing* **438**, 195–208.

Singh, S., Agrawal, S., Sahu, T. & Das, D. 2021 iPipe: water pipeline monitoring and leakage detection. In: *Proceedings of the 2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, Jaipur, India, pp. 367–372.

Soldevila, A., Blesa, J., Tornil-Sin, S., Duviella, E., Fernandez-Canti, R. M. & Puig, V. 2016 Leak localization in water distribution networks using a mixed model-based/data-driven approach. *Control Eng. Pract.* **55**, 162–173.

Tariq, S., Bakhtawar, B. & Zayed, T. 2022 Data-driven application of MEMS-based accelerometers for leak detection in water distribution networks. *Sci. Total Environ.* **809**, 151110.

Weyns, M., Mazaev, G., Vancoillie, F., Vaes, G., De Turck, F., Van Hoecke, S. & Ongenae, F. 2022 A data-driven approach to leak localization in water distribution networks using GIS-enhanced autoencoders. *Urban Water*. (in submission).

Xie, X., Zhou, Q., Hou, D. & Zhang, H. 2018 Compressed sensing based optimal sensor placement for leak localization in water distribution networks. *J. Hydroinf.* **20** (6), 1286–1295.

First received 13 September 2022; accepted in revised form 19 March 2023. Available online 31 March 2023