A Scalable and Dependable Data Analytics Platform for Water Infrastructure Monitoring

Felix Lorenz*, Morgan Geldenhuys*, Harald Sommer†, Frauke Jakobs†,
Carsten Lüring§, Volker Skwarek‡, Ilja Behnke* and Lauritz Thamsen*
*Technische Universität Berlin, Germany, {firstname.lastname}@tu-berlin.de
†Ingenieurgesellschaft Prof. Dr. Sieker mbH, Germany, {initial.lastname}@sieker.de
‡Hochschule für Angewandte Wissenschaften Hamburg, Germany, volker.skwarek@haw-hamburg.de
§WALTER tecyard GmbH & Co. KG, Germany, carsten.luering@walter-tecyard.com

***Abstract*—With weather becoming more extreme both in terms of longer dry periods and more severe rain events, municipal water networks are increasingly under pressure. The effects include damages to the pipes, flash floods on the streets and combined sewer overflows. Retrofitting underground infrastructure is very expensive, thus water infrastructure operators are increasingly looking to deploy IoT solutions that promise to alleviate the problems at a fraction of the cost.**

In this paper, we report on preliminary results from an ongoing joint research project, specifically on the design and evaluation of its data analytics platform. The overall system consists of energy-efficient sensor nodes that send their observations to a stream processing engine, which analyzes and enriches the data and transmits the results to a GIS-based frontend. As the proposed solution is designed to monitor large and critical infrastructures of cities, several non-functional requirements such as scalability, responsiveness and dependability are factored into the system architecture. We present a scalable stream processing platform and its integration with the other components, as well as the algorithms used for data processing. We discuss significant challenges and design decisions, introduce an efficient data enrichment procedure and present empirical results to validate the compliance with the target requirements. The entire code for deploying our platform and running the data enrichment jobs is made publicly available with this paper.

***Index Terms*—Water Networks, Internet of Things, Critical Infrastructures, Predictive Maintenance, Cloud Computing**

I. Introduction

Municipalities around the world use water networks to distribute fresh water and remove sewage from private households and industrial facilities. Their history reaches back into the dawn of civilization and they are generally understood as an essential constituent of our increasingly urbanized society. Today, water networks are unanimously considered to be *critical infrastructure*, because the lives and livelihood of urban populations directly depend on their functioning. Due to the very high cost of directly accessing the infrastructure for upgrades and repair, pipes have been aging under the ground since they were first put into place decades or even centuries ago. Meanwhile, the pressures excerted on the systems from extreme weather events such as prolonged dry periods and flash floods present an increasing threat to their seamless operation [1]. For example, storm drains in the streets can clog, causing the streets to be flooded as seen during the heavy rain events in Berlin in 2017 and Washington DC in 2019. Furthermore, pipes can crack and the water leak out into the ground, which can cause significant harm to the surrounding infrastructures as well as economic damage due to the loss of fresh water. Finally, in the case of a storm event, the load on the network can exceed its capacity and spill untreated into nearby waterways. Such problems could likely be alleviated through large scale real-time measurements of key observables such as pressures, flow rates, temperatures [2]. This information can help to determine the areas of greatest concern, schedule predictive maintenance, and ultimately control the various pumps and reservoirs more optimally. Wireless Sensor Networks (WSNs) as powered by Internet of Things (IoT) technology present a promising tool for this purpose, because they can be deployed in parts of the network that are inaccessible to humans, stay operational over long periods without external power supply and transmit their sensor readings wirelessly to the cloud for further analysis. In this paper, we report on WaterGridSense4.0 (WGS4.0), a joint project involving academic, industrial and municipal partners, that aims to develop an integrated solution based on power- and cost-efficient sensor devices as well as a scalable and fault tolerant data analytics platform. The main contributions of our work can be summarized as follows:

1) We discuss how the integration of the architecture components is subject to a tradeoff between protocol compliance and scalability.
2) We propose an efficient scheme for enriching the stream of sensor measurments with values from a changing set of attributes and evaluate its throughput and latency for different cluster sizes and number of attributes.
3) We report a series of key insights from working on the algorithms addressing three of the many use cases in the WaterGridSense4.0 project.
4) We provide all code required for deploying our platform in kubernetes with configurable cluster size, so our solution can be rolled out for arbitrarily large water networks around the world.

The rest of this paper is structured as follows. In the following section, we review previous efforts to leverage IoT technology for smart water grid monitoring. Section III, describes our system architecture. The subsequent Section IV

reports on measurements with respect to several non-functional requirements. Finally, we discuss uses cases and relevant algorithms in Section V and conclude our work in Section VI.

II. Related Work

With the recent proliferation of small, inexpensive wireless sensor devices with long battery life, a wide range of research projects have been initiated on the potential of IoT technologies for various tasks in urban environments [3]. Many focus specifically on monitoring and control of water grids [4], a direction in the literature often referred to as *smart water grid*. We take a look at the state of the art in this field throughout the remainder of this chapter. It should be noted that we only consider recent approaches based on WSNs technology to keep the discussion within the scope of our own work.

Among the seminal works on this subject is the *PipeNet* system by Stoianov et al. [5]. They discuss an end-to-end architecture for WSN-based pipe monitoring and leakage detection based on the Intel Mote platform. The end device with multiple sensors is installed in a manhole and transmits the readings to the nearest gateway via Bluetooth which in turn relays them to an analytics backend via GPRS. Despite their results being very relevant for our work, their research is a bit dated and uses a completely different technology stack from what is available today. Similar approaches that rely on stationary sensors installed inside or around pipes include *WaterWiSe@SG* [6], *MISE-PIPE* [7], and *SWATS* [8]. With the exception of MISE-PIPE, they all rely on some form of wireless transmission and focus on the development of the end device with the aim of detecting leaks and blockages. Other publications focus more on the analysis of the data acquired from various meters installed across the district to detect anomalous behavior inside the local grid [9], [10].

Another corpus of works considers mobile sensor nodes that move either actively or float passively through the pipes. Among the earliest systems developed in this context is the modular *MAKRO* robot for pipe inspection developed by Scholl et al. [11]. It moves actively through the network and stores the recorded data locally to be retrieved via cable transfer after the robot has been extracted. Similar principles are applied by the *KANTARO* probe [12] that combines data from multiple sensors to obtain an even more detailed picture of the pipe condition. Examples of passively floating probes include the *SewerSnort* system for gas monitoring [13] and the damage-detection probe *SPAMMS* [14]. All store their recorded observations locally (no wireless transmission) but differ in their approaches to in-pipe localization, a topic that we touch upon in Section V-C in our paper. A more advanced scheme is presented in [15], where the authors propose *TriopusNet*, a swarm-like WSN consisting of multiple probes that autonomously (re-)position inside the pipe network. They are mainly concerned with the problems of node placement and data routing.

Fig. 1: WaterGridSense4.0 system architecture. Solid lines represent data streams, dashed lines indicate batch data queries.

The plethora of successful deployments of WSN for water network monitoring shows that the approach holds great potential for improved monitoring and control of water grids. However, they all use just a few nodes while a real deployment in a large city would need to integrate the data from thousands of nodes to provide a complete understanding of the network state. Yet, real applications of WSN in critical infrastructures such as water grids are subject to stringent system requirements and introduce practical challenges such as data enrichment and protocol compliance. Exactly these considerations are at the core of our research.

III. System Architecture

The central problem addressed in the WGS4.0 project is the continuous monitoring and analysis of large scale water infrastructures. To that end, we develop a scalable and dependable cloud-based analytics platform that processes sensor measurements in real time. The overall system architecture of WGS4.0 is displayed in Figure 1 and is composed of three main parts:

The *Sensor Platforms* control the sensor nodes that are placed in various locations across the water grid. Their main tasks include power management, recording measurements, and sending them wirelessly to the Analytics Platform. The sensor device developed within the WGS4.0 project are modular, water-proof and can be deployed both as stationary sensors in key locations and as mobile devices floating through the pipes.

The *Analytics Platform* is the central data processing component that is at the core of our research. It receives a stream of measurements from the Sensor Platforms, processes the data and forwards the results to the GIS Plattform.

The *GIS Platform* represents the point of interaction for the grid operators. It receives the streams of processed sensor events and visualizes them on a map of the water network. Additionally, the operators use the interface to send updates for the enrichment attributes to the Analytics Platform and query archived measurement series.

A. The Analytics Platform

The Analytics Platform is the central data processing component within the WGS4.0 architecture. It is composed of a set of interconnected services that are deployed using parametrized helm charts, so the system can be rolled out at arbitrary scale. We made the code used for deployment of

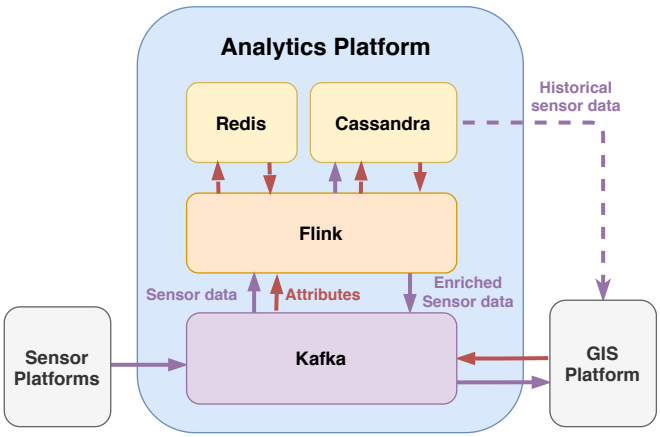

Fig. 2: Architecture of the Analytics Platform.

our plattform available online as open source[1].

A complete schematic of the Analytics Platform is given in Figure 2 and can be summarized as follows: The sensor measurements arrive at the platform through its data stream interface and are picked up by the distributed stream processing engine Apache Flink[2]. Flink then processes the data by subjecting it to the pipelines discussed in Sections III-B and V. The results of the analysis are again streamed to the GIS Platform for live visual display. Additionally, all data is archived in the scalable distributed database Apache Cassandra[3]. The choice and configuration of the individual services is driven by two essential specifications: Requirements and protocols.

a) Requirements: The requirements that our Analytics Platform must meet are defined by the water network operators which in turn must comply with municipal or federal regulations. First and foremost, the system must be *dependable*, i.e. allow continuous operations and produce reliable results even in the presence of partial system failures. We address this issue by introducing service redundancy and using technologies that provide *exactly-once* semantics. Through increasing redundancy, one can achieve almost arbitrary degrees of availability because it is increasingly unlikely that enough workers fail simultaneously to interrupt the service. The second requirement concerns the *responsiveness* of our system, i.e. its end-to-end latency: There should be a low delay between a measurement arriving at the data stream interface and the result of its analysis being sent to the GIS Plattform. Obviously, this delay depends on the data processing jobs, as evaluated in Section IV. Furthermore, in order to apply the WGS4.0 architecture to water networks of different sizes and with varying measurement frequencies, the Analytics Platform must provide *scalability*. To meet this requirement, the deployed services must support increasing the number of worker nodes and the algorithms used in the data processing pipelines must make use of the additional parallelism. Finally, since we want our system to be freely available for municipalities around the world, the developed system must only consist of Free and Open Source Software (FOSS).

b) Integration: Unexpectedly, integration turns out to be a major challenge for the implementation of our Analytics Platform. That is, the data stream interface connecting the three platforms must be at the same time fast, scalable, and compliant with the protocols used by the transmission infrastructure. Within WGS4.0, LoRa has been determined as the most readily available wireless technology among the partnering cities. As is typical in the IoT domain, the various LoRa implementations used by the infrastructure providers in the WGS4.0 project use the Message Queuing Telemetry Transport (MQTT) protocol for passing data upstream. Among the most popular message broker implementations with MQTT support is RabbitMQ[4], which has been shown to achieve up to 40.000 packets per second throughput with a single node [16]. Its scalability is heavily limited by the fact that it does not allow the streams to be partitioned for parallel processing within Apache Flink. On the other hand, Apache Kafka[5] is the fastest event streaming system available today with up to 420.000 packets per second [17] across partitioned streams but it does not natively support MQTT.

Compromises in this tradeoff include using the MQTT connector together with an external broker, relying on the MQTT proxy plugin shipped with the enterprise version of the confluent platform, or, as of recently, using waterstream[6], which merges the two technologies. Unfortunately, the latter two options are not openly available and the first approach introduces a bottleneck, since the MQTT broker has a significantly lower bound on its throughput, as stated above. Within the project, we decided to use Kafka where supported by the local LoRa stack and additionally deploy RabbitMQ to provide MQTT compatibility. Throughout the remainder of this paper, we restrict our evaluation to Apache Kafka since it comes with scalable partitioning of the data streams.

B. Data Enrichment

An important processing step that applies to all considered use cases is *data enrichment*, i.e. the augmentation of the received sensor data with additional information. For example, such information includes geolocations, device type information and measurement units which are not transmitted by the sensors in order to reduce energy consumption. Attribute values are announced and updated via a separate Kafka data stream and stored within the Analytics Platform for use in the enrichment procedure.

Apache Flink supports stateful stream operations with RocksDB as a persistent key/value store, thus enabling fault

[1]https://github.com/dos-group/water-analytics-cluster
[2]https://flink.apache.org/
[3]https://cassandra.apache.org/
[4]https://www.rabbitmq.com/
[5]https://kafka.apache.org/
[6]https://waterstream.io/

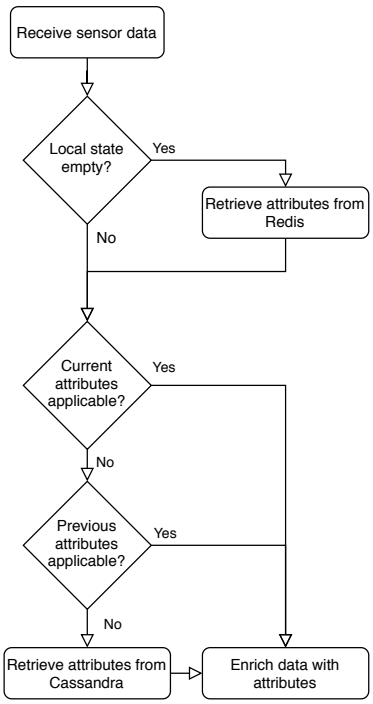

Fig. 3: The complete data enrichment procedure.

tolerant *exactly-once semantics*. Since values stored in Flink managed state can reside on disk instead of memory, we keep them in Java local memory but also store them in Redis[7] and Cassandra. In case of a taskmanager unexpectedly dies, it will load the last checkpoint, replay the data stream and initialize its local state by obtaining the current value for each attribute from Redis.

Due to the high volume of the stream and the lossy nature of LoRa networks, sensor data can arrive delayed and out-of-order. Changes to the enrichment attributes on the other hand are comparatively rare and use reliable transmission which can cause the timestamps of the messages in the sensor data stream to lag behind those of the (already processed) attribute updates. In such cases, the enrichment procedure requires access to previously commissioned attributes to guarantee enrichment with information that was valid at the time when the measurement was taken. At times of high load, this can happen for many measurements, which is why we also store the previous value for each attribute in local memory to avoid costly access to remote storage. In case an out-of-order measurement arrives that is older than both the current and the previous value of a corresponding parameter, we retrieve the correct historical value from a distributed database. The entire sequence of steps performed during data enrichment is presented in Figure 3. The complete source code is available in our git repository[8].

[7]https://redis.io/

[8]https://github.com/dos-group/water-analytics-enrichment

IV. Performance Measurements

Our experimental setup consists of a 30-node Kubernetes cluster co-located with a HDFS cluster of the same size. All nodes run on Intel Xeon E3-1230 V2 CPUs @ 3.30GHz with 16 GB RAM and 1 TB RAID0 HDD (linux sofware raid) and are connected via gigabit ethernet. Each Flink job cluster consists of a single master in high-availability mode and three different cluster sizes taken from $\{4, 8, 12\}$. All Flink workers are created with one task slot and 2 GB of memory. A total of 3 runs of 30 minutes each were conducted for each job configuration.

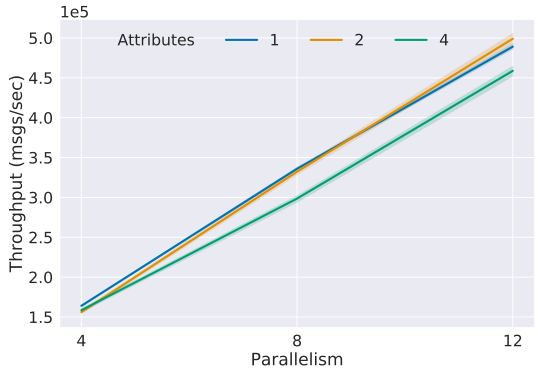

Fig. 4: System throughput in mean number of messages per second for different cluster sizes and number of attributes.

The number of messages processed per second with our pipeline and system configurations are presented in Figure 4. The results show that the data enrichment job achieves stable maximum throughput rates of roughtly 150.000 measurements per 4 taskmanagers and scales linearly with the size of the cluster. The influence that the number of enrichment attributes has on performance seems to become larger with increasing cluster size, which we explain by the non-deterministic partitioning between flink taskmanagers and the redis cluster. Furthermore, the performance difference between enrichment with 1 and 2 attributes is smaller than the residual variance among different measurement runs. Across all configurations, the time it took the operator to "warm up", i.e. fill its local memory with attribute values, was around 45 seconds.

Concerning the requirement of responsiveness, mean operator latencies slightly increase with cluster size, from a median of 151ms (cluster size 4) to 159ms (cluster size 8) and finally 168ms (cluster size 12). While this should be further investigated when considering even larger clusters, we consider these results to be in the range of acceptable values. Overall, our results give conclusive evidence that our implementation is scalable with respect to the rate of incoming sensor data as well as the number of enrichment attributes. In practice, one should configure the system to be at around 50% load during normal operations to use resource efficiently but also give some room for load peaks. For example, a system with cluster size 4 is appropriate for a network with 75.000 sensors that each send one measurement per second.

V. Data Analytics

The flexible design of the sensor nodes developed within WGS4.0 allows deployments across a wide range of locations inside the water infrastructure: It can be attached inside street inlets and manholes but also used as a floating probe for in-pipe inspection tasks. From the many use cases thus enabled, we use this section to present three particularly important ones, report on our investigation into suitable algorithms and discuss some open issues to be addressed during the remainder of the project.

A. Detecting Clogged Storm Drains

Storm drains periodically clog due to leaves and litter that are washed in with the rain, which necessitates periodic cleaning of the internal collection container. Since some are more affected than others, the cost of this maintenance could be significantly reduced if one could remotely estimate the degree of clogging without sending a team for local inspection. The WGS4.0 sensor nodes are intended to be installed within the street inlets to measure the water and dirt levels inside the inlets sludge containment and to estimate the urgency of cleaning them. The cause and impact of clogging on drainage performance has been explored thoroughly [18], [19] and spatiotemporal correlations among nearby street inlets have been shown to exist [20]. Among the proposed solutions is a passive system for estimating water levels in runoffs using RFID tags [21] and a zigbee-based WSN using acoustic sensors [22].

B. Predicting Combined Sewer Overflows

In a combined sewer system, stormwater runoff runs through a single pipe together with wastewater from homes, businesses, and industry. During periods with heavy rainfall, the amount of stormwater can become greater than the network capacity and cause a Combined Sewer Overflow (CSO). This problem could be alleviated by monitoringing the water network in order to predict imminent load spikes and initiate countermeasures, such as preemtive clearing of rainwater tanks and basins. Previous research on CSO detection covers the use of neural networks and control limit theory to detect CSO events from flow measurement time series [23]. In this work, the authors also propose a method to incorporate data from multiple geospatial locations to improve detection. Sonnenberg et al. [24] compare multiple approaches to CSO detection based on water level measurements, rainmeter data and even a simulation model of the water grid.

C. Detecting Leakages and Inflows

Presumably, the use case with the most existing works in the literature concerns the remote detection of leakages and inflow of rainwater or groundwater into wastewater pipes. Many sensor-based solutions have been proposed for this problem, as discussed in Section II. To cover this use case within WGS4.0, sensor nodes are designed to be capable of floating through the pipes while continuously recording key observables, such as temperature and conductivity of the surrounding water. Once they come within range of a gateway, they transmit their measurements to the Analytics Plattform for further inspection. The main difficulty encountered in this use case comes from the fact that radionavigation systems such as GPS don't work deep in the ground and that therefore, the location of each measurement has to be estimated by some other means. Recent solutions to this problem include acoustic localization [25], [26], using gyroscope data [15], [27], and RSSI-based localization with respect to anchor points, e.g. the above mentioned gateways [13], [28].

D. Open Issues

Thorough analysis of the previously proposed solutions for the three use cases reveals a set of required algorithmic features. They include

- the integration of weather data through a separate data stream,
- the support for geospatial queries, and finally
- a scalable solution for detecting anomalous values based on spatial correlations with neighboring sensors

The feasibility of including weather data depends on the products offered by the respective national weather service. In the case of the German DWD, open weather data is made available in the form of daily reports and forecasts that have to be crawled and turned into a data stream to be used within our platform. The second issue concerns the use of geospatial information, e.g. to process readings of physically co-located sensors together. Redis supports geospatial queries by means of the `GEOHASH` and related primitives, but the frequent transmission of query results would likely degrade performance significantly as opposed to performing such computations locally. To fill this gap, the GeoBeam and GeoFlink frameworks have been introduced. They extend the Flink API to enable geospatial queries [29], [30].

Finally, a scalable time series anomaly detection method is needed for all considered use cases, which should provide a means to incorporate information about the geospatial placement of the sensors. There are promising recent results describing a scalable AutoRegressive Integrated Moving Average (ARIMA) implementation in Flink for detecting anomalies in hydrologic time series [31]. Previously, the use of Local Indicators of Spatial Association (LISA) was proposed for anomaly detection in water networks [32]. We are currently working on a pipeline that combines weather- and sensor data from across a certain area and applies spatio-temporal anomaly detection to identify use case-specific target events.

VI. Conclusion

Intelligent monitoring of large scale water infrastructures is an open problem addressed by the WGS4.0 project. As part of its system architecture, we introduce an openly available data analytics platform that runs in the cloud. In this paper, we reported on key insights gained during the development process including component integration, platform design and data analysis. Our platform is shown to meet important critical

infrastructure requirements, specifically dependability, responsiveness and scalability. We further described a complete data enrichment procedure optimized to fit the characteristics of the application domain. In a series of experiments, we demonstrate scalability, quantify responsiveness and provide a point of reference for choosing the right cluster size in any given deployment scenario. Throughout the remainder of the project, we plan to implement and evaluate the outlined analysis algorithms for which certain open issues need to be addressed. These include the integration of weather data, the support for geospatial queries and the use of a scalable anomaly detection method in the stream processor.

ACKNOWLEDGMENTS

This work has been supported through a grant by the German Ministry for Education and Research (BMBF) as WaterGridSense 4.0. We would like to thank all partners involved in this project - Hamburg Wasser, Berliner Wasserbetriebe, Funke Group and ACO Severin Ahlmann GmbH & Co. KG.

REFERENCES

[1] C. Fortier and A. Mailhot, "Climate change impact on combined sewer overflows," *Journal of Water Resources Planning and Management*, vol. 141, no. 5, p. 04014073, 2015.

[2] P. Tsakalides, A. Panousopoulou, G. Tsagkatakis, and L. Montestruque, *Smart Water Grids: A Cyber-Physical Systems Approach*. CRC Press, 2018.

[3] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of network and computer applications*, vol. 60, pp. 192–219, 2016.

[4] J. Dong, G. Wang, H. Yan, J. Xu, and X. Zhang, "A survey of smart water quality monitoring system," *Environmental Science and Pollution Research*, vol. 22, no. 7, pp. 4893–4906, 2015.

[5] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "Pipeneta wireless sensor network for pipeline monitoring," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 264–273.

[6] A. J. Whittle, L. Girod, A. Preis, M. Allen, H. B. Lim, M. Iqbal, S. Srirangarajan, C. Fu, K. J. Wong, and D. Goldsmith, "Waterwise@ sg: A testbed for continuous monitoring of the water distribution system in singapore," in *Water Distribution Systems Analysis 2010*, 2010, pp. 1362–1378.

[7] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz, "Mise-pipe: Magnetic induction-based wireless sensor networks for underground pipeline monitoring," *Ad Hoc Networks*, vol. 9, no. 3, pp. 218–227, 2011.

[8] S. Yoon, W. Ye, J. Heidemann, B. Littlefield, and C. Shahabi, "Swats: Wireless sensor networks for steamflood and waterflood pipeline monitoring," *IEEE network*, vol. 25, no. 1, pp. 50–56, 2011.

[9] L. Perelman, J. Arad, M. Housh, and A. Ostfeld, "Event detection in water distribution systems from multivariate water quality time series," *Environmental science & technology*, vol. 46, no. 15, pp. 8212–8219, 2012.

[10] D. Loureiro, C. Amado, A. Martins, D. Vitorino, A. Mamade, and S. T. Coelho, "Water distribution systems flow monitoring and anomalous event detection: A practical approach," *Urban Water Journal*, vol. 13, no. 3, pp. 242–252, 2016.

[11] K.-U. Scholl, V. Kepplin, K. Berns, and R. Dillmann, "Controlling a multi-joint robot for autonomous sewer inspection," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 1701–1706.

[12] A. A. Nassiraei, Y. Kawamura, A. Ahrary, Y. Mikuriya, and K. Ishii, "Concept and design of a fully autonomous sewer pipe inspection mobile robot" kantaro"," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 136–143.

[13] J. Kim, J. S. Lim, J. Friedman, U. Lee, L. Vieira, D. Rosso, M. Gerla, and M. B. Srivastava, "Sewersnort: A drifting sensor for in-situ sewer gas monitoring," in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 2009, pp. 1–9.

[14] J.-H. Kim, G. Sharma, N. Boudriga, and S. S. Iyengar, "Spamms: A sensor-based pipeline autonomous monitoring and maintenance system," in *2010 Second International Conference on COMmunication Systems and NETworks (COMSNETS 2010)*. IEEE, 2010, pp. 1–10.

[15] T. T.-T. Lai, W.-J. Chen, K.-H. Li, P. Huang, and H.-H. Chu, "Triopusnet: automating wireless sensor network deployment and replacement in pipeline monitoring," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, 2012, pp. 61–72.

[16] P. Dobbelaere and K. S. Esmaili, "Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper," in *Proceedings of the 11th ACM international conference on distributed and event-based systems*, 2017, pp. 227–238.

[17] G. Hesse, C. Matthies, T. Rabl, and M. Uflacker, "How fast can we insert? a performance study of apache kafka," *arXiv preprint arXiv:2003.06452*, 2020.

[18] M. Gómez, G. H. Rabasseda, and B. Russo, "Experimental campaign to determine grated inlet clogging factors in an urban catchment of barcelona," *Urban Water Journal*, vol. 10, no. 1, pp. 50–61, 2013.

[19] M. Rietveld, F. Clemens, and J. Langeveld, "Monitoring and statistical modelling of the solids accumulation rate in gully pots," *Urban Water Journal*, vol. 17, no. 6, pp. 549–559, 2020.

[20] E. S. Pulido, C. V. Arboleda, and J. P. Rodríguez Sánchez, "Study of the spatiotemporal correlation between sediment-related blockage events in the sewer system in bogotá (colombia)," *Water Science and Technology*, vol. 79, no. 9, pp. 1727–1738, 2019.

[21] A. Atojoko, N. Jan, F. Elmgri, R. A. Abd-Alhameed, C. H. See, and J. M. Noras, "Energy efficient gully pot monitoring system using radio frequency identification (rfid)," in *2013 Loughborough Antennas & Propagation Conference (LAPC)*. IEEE, 2013, pp. 333–336.

[22] C. H. See, K. V. Horoshenkov, R. A. Abd-Alhameed, Y. F. Hu, and S. J. Tait, "A low power wireless sensor network for gully pot monitoring in urban catchments," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 1545–1553, 2011.

[23] D. Sumer, J. Gonzalez, and K. Lansey, "Real-time detection of sanitary sewer overflows using neural networks and time series analysis," *Journal of environmental engineering*, vol. 133, no. 4, pp. 353–363, 2007.

[24] H. Sonnenberg, E. Pawlowsky-Reusing, M. Riechel, N. Caradot, E. Toth, A. Matzinger, and P. Rouault, "Different methods of cso identification in sewer systems and receiving waters," in *12th International Conference on Urban Drainage*, 2011.

[25] D. W. Kurtz, "Developments in a free-swimming acoustic leak detection system for water transmission pipelines," in *Pipelines 2006: Service to the Owner*, 2006, pp. 1–8.

[26] D. Kumar, D. Tu, N. Zhu, R. A. Shah, D. Hou, and H. Zhang, "The free-swimming device leakage detection in plastic water-filled pipes through tuning the wavelet transform to the underwater acoustic signals," *Water*, vol. 9, no. 10, p. 731, 2017.

[27] J. Zheng, S. Wang, A. Hazim, and S. Dubljevic, "Pipeline leak detection swimming robot design and deployment," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 1166–1171.

[28] W. Gong, M. A. Suresh, L. Smith, A. Ostfeld, R. Stoleru, A. Rasekh, and M. K. Banks, "Mobile sensor networks for optimal leak and backflow detection and localization in municipal water networks," *Environmental modelling & software*, vol. 80, pp. 306–321, 2016.

[29] Z. He, G. Liu, X. Ma, and Q. Chen, "Geobeam: A distributed computing framework for spatial data," *Computers & Geosciences*, vol. 131, pp. 15–22, 2019.

[30] S. A. Shaikh, K. Mariam, H. Kitagawa, and K.-S. Kim, "Geoflink: A framework for the real-time processing of spatial streams," *arXiv preprint arXiv:2004.03352*, 2020.

[31] F. Ye, Z. Liu, Q. Liu, and Z. Wang, "Hydrologic time series anomaly detection based on flink," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[32] D. E. Difallah, P. Cudre-Mauroux, and S. A. McKenna, "Scalable anomaly detection for smart city infrastructure networks," *IEEE Internet Computing*, vol. 17, no. 6, pp. 39–47, 2013.