Demo: LE3D: A Privacy-preserving Lightweight Data Drift Detection Framework

Ioannis Mavromatis and Aftab Khan
Bristol Research and Innovation Laboratory (BRIL), Toshiba Europe Ltd., Bristol, UK
Emails: {Ioannis.Mavromatis, Aftab.Khan}@toshiba-bril.com

***Abstract*—This paper presents LE3D; a novel data drift detection framework for preserving data integrity and confidentiality. LE3D is a generalisable platform for evaluating novel drift detection mechanisms within the Internet of Things (IoT) sensor deployments. Our framework operates in a distributed manner, preserving data privacy while still being adaptable to new sensors with minimal online reconfiguration. Our framework currently supports multiple drift estimators for time-series IoT data and can easily be extended to accommodate new data types and drift detection mechanisms. This demo will illustrate the functionality of LE3D under a real-world-like scenario.**

***Index Terms*—Data Drift, IoT, Drift Detector, Resource-Constrained, Ensemble Learning**

I. Introduction and Motivation

IoT sensors are found in numerous domains, e.g., air pollution monitoring, farming, smart cities, etc. [1]. These applications rely on data fidelity. Considering the scale and economic viability, the use of low-cost sensors is inevitable. However, the low-cost nature of these sensors, the differences between manufacturers, the lack of reliable calibration, and the "silicon gamble", can lead to inconsistencies. The differences become even more prominent when data from different devices are compared [2], making the data's relative measurements the only viable strategy for comparison between them.

Moreover, the data integrity in the above applications can be of paramount importance [3]. Altered data streams, either from benign cases (e.g., faulty sensors) or malicious actions (e.g., unauthorised data tampering), can disrupt or bias an application and result in widespread damage and outages. Therefore, preserving data privacy while ensuring their integrity has recently become a big topic of scientific discussion, with various data drift solutions being proposed, e.g., [4]–[6].

Previous activities either focused on improving the prediction accuracy, operated without considering data privacy (where the data is stored and processed), or without provision for real-world deployment. Inspired by the above, we present *Lightweight Ensemble of Data Drift Detectors (LE3D)*, a novel lightweight data drift detection framework. Its operation is two-fold, i.e., it can be used: *1)* for evaluating novel data drift detection strategies, and *2)* for real-world deployments detecting different types of drift in multiple sensors. Our framework is publicly available at `github.com/toshiba-bril/le3dDataDriftDetector`.

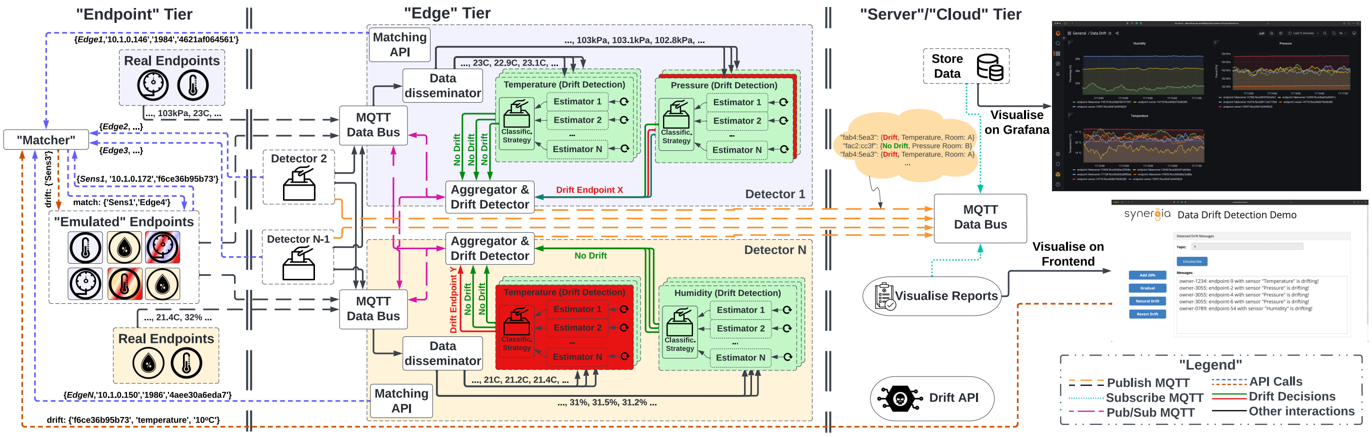
II. LE3D: Main System Components

LE3D is designed with both research and real-world scenarios - consumer applications in mind. It consists of an extensible drift detection framework and a set of supporting tools. Within LE3D, an *estimator* can be a statistical or a Machine Learning process that classifies a sensor sample as drifting or not. The *estimators* can operate in an *online learning*-fashion, adapting to the data distribution changes or working as static classifiers (e.g., thresholding methods). LE3D supports various statistical drift estimators, i.e., ADaptive WINdowing (ADWIN), Page-Hinkley Test (PHT), and Kolmogorov-Smirnov Windowing (KSWIN) for time series data [7]. These estimators or the data fed into them can be easily replaced or extended as required.

A *detector* plays multiple roles. Firstly, it handles received data streams. Later, based on the data type, it can assign one or many *estimators* for each sensor stream. Ensemble strategies can also be introduced to enhance the classification accuracy (e.g., voting mechanisms taking into account windows of samples and multiple *estimators*' outcome). Finally, if required by an end-user, a *detector* can relay the sensor data to the backend for visualisation, storage, and demonstration purposes.

LE3D ensures data privacy. All decisions are taken at the "edge" without data being exchanged across the network. Multiple *detectors* can collectively classify the detected drift type as *natural* – happening concurrently across multiple sensors – or *abnormal* – only one sensor is drifting. This is performed through an *aggregator* that operates in parallel with each *detector*. The *aggregators* share only the classification decision, the metadata provided by an endpoint, and the result of a one-sample K-S test, with the surrounding devices preserving the data privacy. More details about the algorithms and intelligence introduced within LE3D can be found in [8].

LE3D comes with a set of supporting tools for visualisation and experimentation. As access to real-world drifting endpoints is not always possible, LE3D provides a *streamer* and an *emulator*. The *streamer* streams "real-world" data from a pre-existing dataset (in CSV format). On the other hand, an *emulator* generates "realistic" emulated data streams and drifts on demand. Their statistical properties can be based on real data to ensure realistically emulated sensors. Moreover, a *matching* framework ensures the system's scalability when large-scale experiments are conducted. This framework operates as an oracle and associates different endpoints, *emulators*, *streamers*, and *detectors* for large-scale experimentation. Finally, LE3D comes with a simple GUI for visualising and controlling the drifts introduced. The GUI can be easily extended to support more drifts or visualisation interfaces.



owner-1234: endpoint-9 with sensor "Temperature" is drifting!
owner-3055: endpoint-2 with sensor "Pressure" is drifting!
owner-3055: endpoint-4 with sensor "Pressure" is drifting!
owner-3055: endpoint-6 with sensor "Pressure" is drifting!
owner-0789: endpoint-54 with sensor "Humidity" is drifting!

Fig. 1. A detailed system diagram visualising the different system components and the interactions between them. LE3D operates in a three-tier architecture, with each component running as a microservice. All interactions occur either via well-defined MQTT topics and messages or RESTful APIs.

III. LE3D: System Architecture/Implementation

LE3D operates in a standard three-tier architecture (Fig. 1): cloud, edge, and endpoint. The "cloud" hosts the frontend, the *matching* framework, and a database for storage. The "edge" tier hosts the *detectors*, the data messaging bus, the *aggregators*, and is responsible for sharing the results with the "cloud". Finally, the "endpoints" can be either real or emulated sensor streams incorporating no intelligence.

All the different components are independent microservices that can be deployed and updated on demand. LE3D is highly scalable and extensible and allows the end-user to implement new functions within the existing applications or replace them entirely with new ones (e.g., replace the frontend GUI with a different application). LE3D can be deployed locally for testing (on a single computer) or across a distributed Kubernetes cluster. At its minimum configuration, LE3D requires just a single *detector* on each "edge" device and a way to receive sensor data from one or multiple endpoints. Our system architecture diagram can be found in Fig. 1

The communication plane is based on MQTT and RESTful APIs. All the sensor data and decisions are exchanged via multiple MQTT predefined topics. More specifically, the detector decisions are published as retained messages, while the sensor samples as regular ones. Moreover, the interactions between the applications (e.g., the association of emulators to detectors) are done via RESTful APIs.

The default configuration can be overridden with environmental variables during the execution. All the above results in a very flexible, robust, and extensible implementation that can be used either for research-driven drift detection activities or deployment in realistic scenarios. Moreover, the core functionality of LE3D introduces minimal overhead and can be scaled up across tens or hundreds of devices.

Our framework was implemented in Python 3.9.12. The existing estimators' functionality is based on River online/streaming ML package [9]. All statistical calculations and optimisation mechanisms are based on SciPy and NumPy libraries. The MQTT messaging relies on Eclipse's Paho MQTT client implementation. The frontend and all the RESTful APIs are developed with Flask. The functionality of the different components described in Sec. II was developed in-house. Finally, the *detector*'s and *aggregator*'s functionality has been tested on a Raspberry Pi (RPi) Compute Module 3b+ [8], with a BCM2837B0 Cortex-A53 64-bit 1.2 GHz System-on-a-Chip (SoC) and 1 GB of RAM. This RPi was chosen as a representative resource-constrained IoT device.

IV. Demonstration

The demonstration will showcase the functionality of LE3D in a real-world-like scenario. Various emulated endpoints and streamers will be executed, generating realistic traffic targeting multiple detectors. An end-user will introduce various drifts, which will be identified by a lightweight ensemble data drift detection implementation running in a distributed fashion. Finally, all the results will be displayed to the end user.

References

[1] H. Arasteh, V. Hosseinnezhad *et al.*, "IoT-based Smart Cities: A Survey," in *Proc. of IEEE EEEIC 2016*, Jun. 2016, pp. 1–6.

[2] P. Ferrer-Cid, J. M. Barcelo-Ordinas *et al.*, "A Comparative Study of Calibration Methods for Low-Cost Ozone Sensors in IoT Platforms," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9563–9571, Jul. 2019.

[3] M. N. Aman, B. Sikdar *et al.*, "Low Power Data Integrity in IoT Systems," *IEEE Internet Things J.*, vol. 5, no. 4, May 2018.

[4] O. A. Wahab, "Intrusion Detection in the IoT under Data and Concept Drifts: Online Deep Learning Approach," *IEEE Internet Things J.*, pp. 1–1, Apr. 2022.

[5] B. Friedrich, T. Sawabe *et al.*, "Unsupervised Statistical Concept Drift Detection for Behaviour Abnormality Detection," *Applied Intelligence*, vol. 58, no. 3, pp. 509–523, May 2022.

[6] L. Yang, D. M. Manias *et al.*, "PWPAE: An Ensemble Framework for Concept Drift Adaptation in IoT Data Streams," in *Proc. of IEEE GLOBECOM 2021*, Dec. 2021, pp. 01–06.

[7] F. Bayram, B. S. Ahmed *et al.*, "From Concept Drift to Model Degradation: An Overview on Performance-aware Drift Detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, Mar. 2022.

[8] I. Mavromatis, A. Sánchez-Mompó *et al.*, "LE3D: A Lightweight Ensemble Framework of Data Drift Detectors for Resource-Constrained Devices," *arXiv:2211.01840 [cs.LG]*, Jan. 2023.

[9] J. Montiel, M. Halford *et al.*, "River: Machine Learning for Streaming Data in Python," *J. Mach. Learn. Res.*, vol. 22, no. 1, Jul. 2022.