Real-Time Anomaly Detection in IoT Sensor Streams Using Kafka Streams
Use-Case Specification

Mihalache Mihai-Andrei, Darius-Ștefan Jurjiu-Țândrău, Andrei Moldovan, Vlad Strilețchi

1. Problem / Motivation

Large-scale IoT deployments (industry, smart buildings, energy grids) generate continuous high-velocity sensor data. Manual monitoring or simple threshold rules cannot reliably detect unexpected or subtle anomalies, such as equipment malfunction, abnormal consumption, or security incidents. The challenge is to build a scalable, real-time, automated anomaly detection pipeline that can process thousands of events per second with low latency.

2. Use Case Definition

We propose an end-to-end big-data solution for real-time anomaly detection in IoT sensor streams. Data from simulated IoT sensors (e.g., temperature, pressure, vibration, energy metrics) will be ingested into a distributed streaming system. The goal is to automatically identify abnormal behavior and generate alerts while maintaining scalability, fault tolerance, and near-real-time responsiveness.

3. Data Characteristics

- Source: Python-based IoT simulators publishing to Kafka.
- Format: JSON records containing sensor_id, timestamp, and sensor metrics.
- Volume: Configurable to emulate realistic loads (1k–50k events/sec).
- Ground Truth: Unsupervised setup; anomalies injected into simulations for evaluation.

4. Objectives

1. Ingest sensor streams using Kafka.
2. Process and aggregate data in real time using Kafka Streams.
3. Apply machine-learning-based anomaly detection (PyTorch / scikit-learn).
4. Persist raw and processed data in Hadoop (HDFS) for offline training and analysis.
5. Produce anomaly alerts and provide a simple visualization/demo.

5. Proposed Architecture (High-Level)

- Ingestion: IoT simulators → Kafka topics (sensors.raw).

- Stream Processing: Kafka Streams for cleaning, windowing, and feature extraction → write to sensors.features.
- Anomaly Detection Service: Python microservice consumes features and computes anomaly scores using ML models (e.g., isolation forest or autoencoder) → outputs to sensors.anomalies.
- Storage: Kafka Connect / custom consumers write data to HDFS for historical analysis.
- Dashboard / Demo: Minimal UI or notebook showing sensor values, anomaly alerts, and system metrics.

6. Research Basis & Expected Outcomes

The solution draws on current research in streaming anomaly detection and scalable event-processing architectures. We will compare at least two ML algorithms and evaluate accuracy, throughput, and latency. The final prototype will demonstrate real-time anomaly detection under simulated sensor loads, showing the feasibility of scalable automated monitoring in IoT environments.