Real-Time Fraud Detection in Mobile Money Systems Using an Edge–Fog–Cloud Streaming Architecture

Cheikh Bay Oudaa [C16706]
Department of Computer Science, Master 2 AI, ML & DS
University of Nouakchott
Supervisor: Dr. El Benany Med Mahmoud
February 2026

Abstract

Mobile money services have become a cornerstone of financial transactions in Mauritania, enabling fast and accessible payments across the country. However, this rapid growth has been accompanied by an increase in fraudulent activities, requiring efficient and scalable fraud detection mechanisms. Traditional centralized fraud detection systems suffer from high latency, limited scalability, and privacy concerns. In this work, we propose a real-time distributed fraud detection architecture based on Edge–Fog–Cloud computing, leveraging Apache Kafka for data streaming and Python-based analytics. The proposed architecture demonstrates the feasibility of near real-time fraud monitoring, system modularity, and scalability, making it suitable for mobile money ecosystems in developing countries.

Keywords: Fraud Detection, Mobile Money, Edge Computing, Apache Kafka, Streaming Systems, Cloud Analytics

1 Introduction

Mobile money platforms such as Bankily and Masrivi play a crucial role in Mauritania's financial inclusion. Millions of transactions are processed daily, making fraud detection a critical challenge for financial institutions. Conventional fraud detection solutions rely on centralized systems where all transaction data are collected and processed in a single location. While effective in some contexts, this approach introduces several limitations: high communication latency, poor scalability under heavy transaction loads, and increased risk of data exposure.

To address these challenges, distributed computing paradigms such as Edge Computing and streaming architectures have gained increasing attention. This project proposes a distributed Edge–Fog–Cloud architecture for fraud detection in mobile money systems. Instead of sending all data directly to a central server, transactions are produced at the Edge, streamed via a Fog layer using Apache Kafka, and analyzed and visualized at the Cloud level.

2 Methodology

2.1 System Architecture

The proposed system is structured into three main layers:

Edge Layer – Transaction Generation
The Edge layer simulates mobile money agencies or transaction sources. Transactions are generated using Python scripts. Each transaction includes amount, agency, timestamp, and fraud label. Data generation mimics real-world transaction flows. This layer represents geographically distributed mobile money agents.

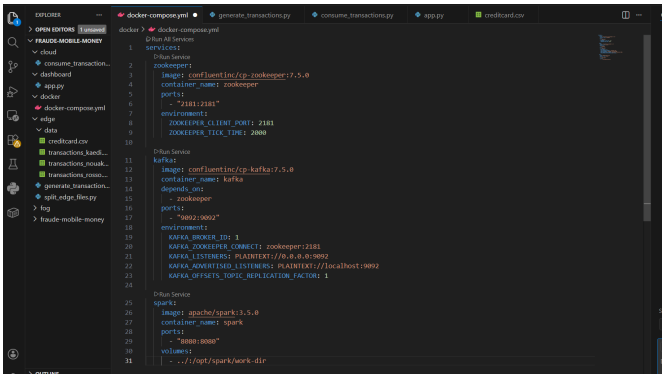

Figure 1: Figure 1: Implementation of the Edge layer transaction generation scripts in VS Code.

Fog Layer – Streaming and Messaging
The Fog layer is implemented using Apache Kafka. Kafka acts as a message broker. Edge nodes publish transactions as producers. Data are streamed asynchronously to consumers. Kafka ensures fault tolerance and decoupling between components. This layer allows the system to handle high-throughput transaction streams efficiently.

Cloud Layer – Monitoring and Visualization
The Cloud layer is responsible for consuming transaction streams from Kafka, aggregating transaction data, and displaying results in real time using Streamlit. A dashboard provides insights such as: number of transactions, fraud occurrences, distribution by agency, and transaction history.

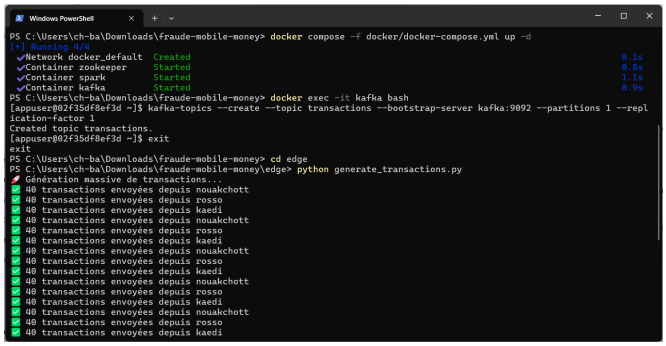

Figure 2: Figure 2: Fog layer activity monitored via Kafka broker streaming logs.

2.2 Fraud Labeling Logic

Fraud detection in this project is simulated using rule-based and probabilistic logic: most transactions are labeled as legitimate. Fraud labels (1, 2, or higher) are introduced randomly. This approach allows testing of the pipeline without exposing real financial data.

3 Experimental Setup

The system was deployed on a local machine using Docker containers. Configuration includes: Edge (Python scripts generating transactions), Fog (Apache Kafka + Zookeeper), and Cloud (Streamlit dashboard). The operating system used is Windows, with tools including Python, Kafka-Python, Docker, Pandas, and Streamlit. Transactions are continuously streamed from the Edge to Kafka and consumed by the Cloud dashboard.

4 Results

The system successfully streamed and visualized mobile money transactions in near real time. Observed results include: continuous transaction flow from Edge to Kafka, real-time update of dashboard metrics, clear visualization of fraud labels, and the ability to scale the number of generated transactions. The architecture demonstrated stable performance under increasing transaction volume, confirming the effectiveness of Kafka as a Fog-layer streaming solution.

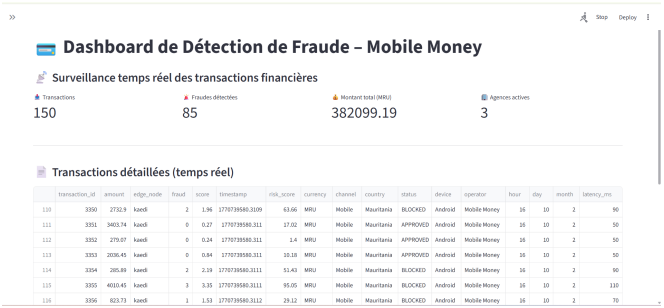

Figure 3: Figure 3: Real-time Cloud Dashboard providing fraud analytics and monitoring.

5 Discussion

The results highlight several advantages of the proposed architecture: low latency (transactions are processed and visualized almost instantly), scalability (Kafka enables handling large streams of data), and modularity (Edge, Fog, and Cloud layers are loosely coupled). Security awareness is maintained as no real customer data are required. Although the fraud detection logic is currently rule-based, the architecture is compatible with future integration of machine learning or deep learning models.

6 Conclusion and Future Work

This project demonstrates a functional prototype of a distributed fraud detection system using an Edge–Fog–Cloud architecture. The use of Apache Kafka enables reliable real-time streaming, while Streamlit provides an intuitive monitoring interface. Future work includes: integrating machine learning models (Spark ML / Scikit-learn), adding real-time fraud scoring, and deploying on cloud infrastructure with enhanced security.

References

[1] Apache Software Foundation, Apache Kafka Documentation, 2024.

[2] Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," CACM, 2016.

[3] Python Software Foundation, Python Documentation, 2024.

[4] Streamlit Inc., Streamlit Documentation, 2024.