Federated Analytics Hybrid Architecture for Connected Truck Data Analysis

Aslihan Reyhanoglu[1], Feyzi Ege Kumec[1], Abdulkadir Bilge[1],
Hira Bakirhan[2], Bugra Turan[3], and Sinem Coleri [3]
[1]Koc University Ford Otosan Automotive Technologies Laboratory (KUFOTAL), Sariyer, Istanbul, Turkey, 34450
[2]Ford Otosan, Sancaktepe, Istanbul, Turkey, 34885
[3]Department of Electrical and Electronics Engineering, Koc University, Sariyer, Istanbul, Turkey, 34450
E-mail: [areyhanoglu, fkumec, abilge20, buturan, scoleri] @ku.edu.tr and hertugru@ford.com.tr

Abstract*—Connected vehicles offer opportunities to optimize fleet operations by leveraging data from their on-board sensors. Off-board cloud architectures play a key role in enhancing fleet-wide health monitoring and operational efficiency. Federated Analytics (FA) empowers connected vehicles to enable decentralized data analysis by improving privacy, decreasing cloud data transmission, and supporting real-time decision-making. This paper contributes a practically deployable federated analytics-assisted edge–cloud architecture for connected truck health monitoring, where high-rate telemetry is processed locally on an NVIDIA Jetson Orin and only compact event summaries are transmitted to the cloud for fleet-level reasoning. We further introduce a cloud-side Temporal-Spatial Aggregation (TSA) mechanism that computes near real-time anomaly hotspots using 15-minute sliding windows over geo-fenced zones, enabling explainable time- and location-correlated insights. Finally, we demonstrate feasibility with real truck data and provide an End-to-End (E2E) latency and Uplink (UL) data volume reduction against a raw data transfer baseline.

***Keywords*— *Federated Analytics*; *Edge Computing*; *Connected Trucks*; *Temporal–Spatial Aggregation*; *Fleet Health Monitoring*.**

I. Introduction

Connected vehicles offer opportunities to enhance safety and performance using data-driven insights from on-board sensors for fleet management. Complex data and heavy transmission demands can overwhelm centralized systems, leading to high latency, network congestion, and costly cloud storage. 5G Automotive Association (5GAA) defines for remote vehicle health monitoring, a latency of less than 30 seconds [1] is acceptable, allowing periodic updates without compromising safety. On the other hand, 5GAA [2] specifies a latency of 100 milliseconds to support data sharing use cases of dynamic objects. To address these latency requirements, technologies, such as FA and Multi-Access Edge Computing (MEC) are provisioned to be integrated into Vehicle-to-Network-to-Vehicle (V2N2V) schemes [3]. V2N2V systems enable seamless communication between vehicles and the cloud, supporting real-time decision-making [4]. V2N2V use cases aim to support cooperative collision avoidance and driving, together with fleet management use cases, requiring low latency and high reliability [5]. Unlike Federated Learning (FL), which primarily exchanges model updates to collaboratively train a global model, FA focuses on decentralized data analysis where raw data remains on the vehicles and only compact analytical outputs (events/aggregates) are shared. FA addresses latency, scalability, and privacy challenges in connected vehicle systems by combining localized edge computing with cloud-based aggregated data analysis [6]. Edge computing enables real-time event detection whereas reducing cloud dependency, latency, and resource usage.

In the literature, significant advancements have been proposed to improve the reliability, latency, and efficiency of connected vehicle systems through V2N2V communications [5], federated analytics [6]–[8], and edge computing [9], [10]. In [5], an analytical model incorporating numerology, scheduling, and Hybrid Automatic Repeat Request (HARQ) evaluates 5G New Radio (NR) V2N2V configurations to meet the 6 ms UL+Downlink (DL) radio-latency budget derived from the 3rd Generation Partnership Project (3GPP) 10 ms E2E target for High Level of Automation (HLoA) cooperative lane change. The decentralized federated analytics framework in [6] converges with fewer communication rounds and reduces communication cost by up to 89% in a simulated Vehicular Ad Hoc Network (VANET) with Vehicle-to-Vehicle (V2V) neighbor broadcast. Similarly, [7] introduces a federated anomaly detection approach that filters malicious updates without centralized data, achieving up to 6.9× higher accuracy under strong attacks and offering formal robustness guarantees. In [8], a privacy-preserving framework enables vehicles to transmit locally processed analytics insights, such as object detection results to nearby MEC/Road Side Unit (RSU)s. These edge nodes aggregate the data and broadcast updated High Definition (HD) map tiles. A two-tier 5G edge infrastructure, optimized using queuing theory, achieves optimized latency for map updates in dense urban environments, as demonstrated through Simulation of Urban MObility (SUMO) simulations. [9] shows that in realistic multi-Mobile Network Operator (MNO) 5G V2N2V scenarios, edge-proximal MEC deployments with local peering can keep the average E2E latency below 20 ms for a wide range of loads, indicating feasibility for time-critical cooperative maneuvers such as lane merging. [10] shows that MEC federation can reduce E2E latency by over 80% in multi-MNO 5G V2N2V scenarios, supporting the feasibility of sub-20 ms cooperative maneuvers such as lane merging. However, none of the studies to date addressed real-time truck health monitoring using processed anomalies without heavy on-board retraining. By combining low-latency edge detection with cloud-based temporal–spatial aggregation, we offer a scalable solution for fleet-wide monitoring.

This paper presents a FA-assisted edge–cloud architecture for connected trucks, where raw sensor data is processed on NVIDIA Jetson Orin device to generate lightweight event

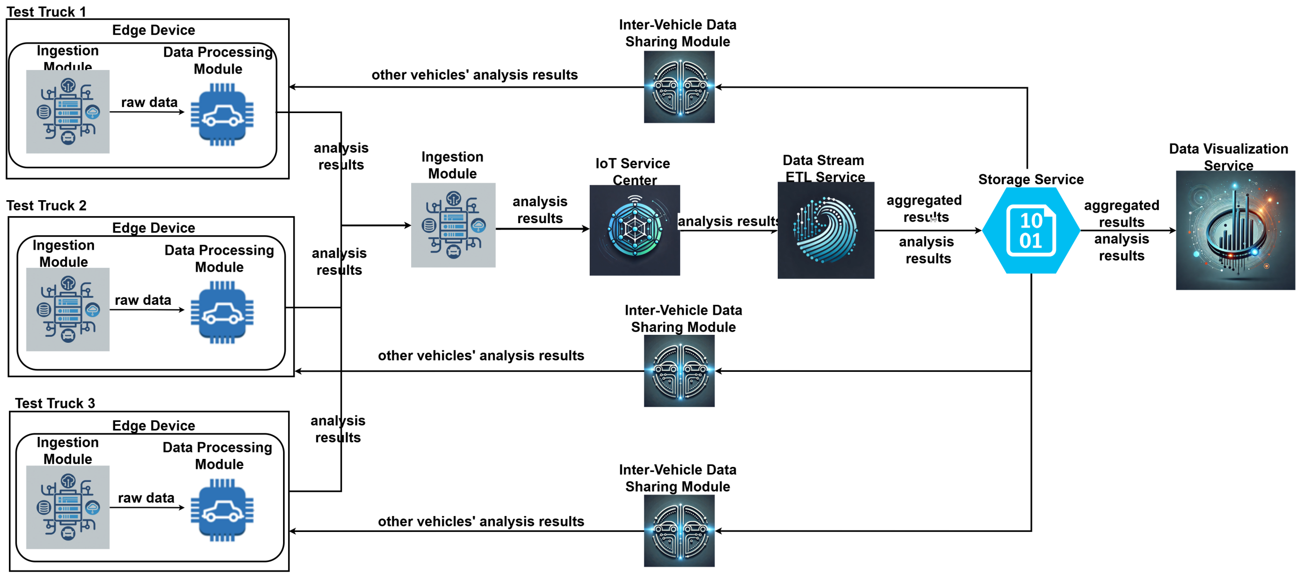

Figure 1. Proposed Federated Analytics Architecture (FA Architecture).

summaries for real-time truck health monitoring. We also introduce cloud-based TSA, using 15-minute time windows over geo-fenced zones to create real-time anomaly hotspot maps. We demonstrate efficient data reduction and achieve approximately 1-second E2E latency by using real data from Ford F-Max test trucks and synthetic data.

The rest of the paper is organized as follows. Section II presents the system model. Section III describes the algorithms for analysis and aggregation. Section IV provides benchmarking and performance evaluation results, focusing on end-to-end latency metrics and comparisons with the raw data transfer baseline. Finally, Section V concludes the paper and outlines future work.

II. System Model

This section presents the proposed FA Architecture (see Figure 1) for scalable, secure, and real-time truck behavior and health monitoring in connected fleet systems. It supports continuous data collection, processing, storage, and visualization across on-board and off-board systems. The architecture includes four main components: the Ingestion Module, Data Processing Module, Off-Board Backend, and Inter-Vehicle Data Sharing Module. On-board modules run on an NVIDIA Jetson Orin System-on-Chip (SoC), enabling real-time edge processing. The Off-Board Backend handles long-term storage, data exchange, and visualization through its Internet of Things (IoT) Service Center, Extract-Transform-Load (ETL) Service, Storage Service, and Visualization Service. Table I summarizes the modules and their relationship to FA.

The Ingestion Module (see Figure 2) acts as the communication bridge between edge devices and the cloud. It collects raw sensor data via a Robot Operating System (ROS) topic consumer and forwards these messages to the on-board Data Processing Module. In addition, it transmits the processed results to the cloud in real time, enabling bidirectional communication and

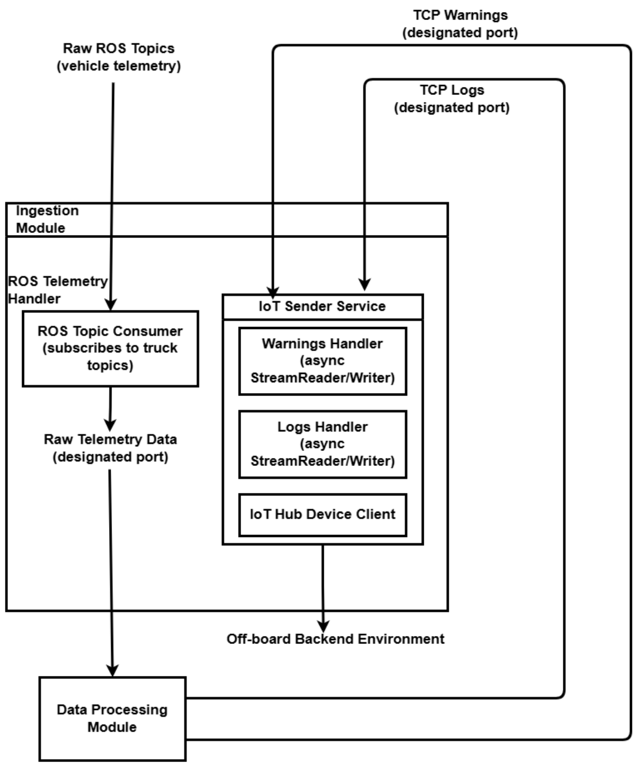
TCP Warnings (designated port)
Raw ROS Topics (vehicle telemetry)
TCP Logs (designated port)
Ingestion Module
ROS Telemetry Handler
ROS Topic Consumer (subscribes to truck topics)
IoT Sender Service
Warnings Handler (async StreamReader/Writer)
Logs Handler (async StreamReader/Writer)
IoT Hub Device Client
Raw Telemetry Data (designated port)
Off-board Backend Environment
Data Processing Module

Figure 2. Architecture of the ingestion module.

scalable connection management. Each truck is registered as a distinct IoT entity, which enhances security and access control.

The Data Processing Module serves as the on-board analytics engine, processing high-frequency telemetry to detect anomalies, such as harsh braking, aggressive acceleration, steering issues, and sensor faults. It uses socket connections to stream data and applies thresholds derived from domain-specific heuristics to identify abnormal behavior.

The IoT Service Center enables bidirectional communication by facilitating devices to transmit telemetry data to the cloud and also receive updates and commands from the cloud [11]. In our context, it plays a role as a secure and scalable bridge between data transfer module and cloud services. We create three separate devices on the IoT Service Center to manage and monitor each vehicle's data individually. Each vehicle has its own identity within the IoT system, which enhances organization, security, and scalability in the IoT architecture.

The Data Stream ETL Service aggregates telemetry and anomaly events (e.g., throttle, brake, GPS, steering, sensor health) using 15-minute sliding windows with 1-minute hops. It streams edge-processed insights in real time to the Storage Service [12].

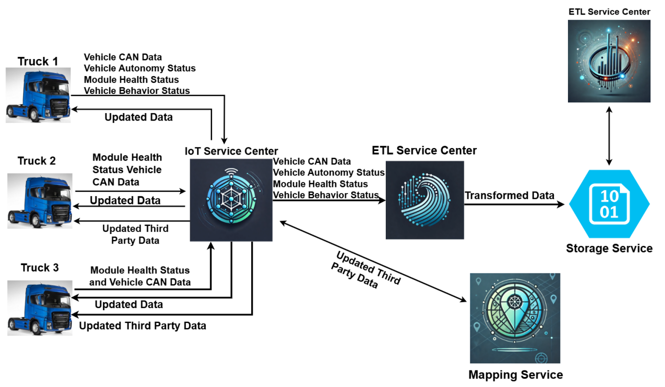

Figure 3. An Architecture for Raw Data Transfer.

The Storage Service provides long-term storage for analysis results and aggregated data, organized hierarchically by vehicle and timestamp. This structure improves retrieval efficiency and reduces operational costs [13].

The Data Visualization Service enables real-time analytics and querying through external tables, avoiding the need for data import. Dashboards show module health, warnings, mode changes, and system metrics for both real-time and historical analysis.

The Inter-Vehicle Data Sharing Module allows connected trucks to exchange processed data in real time. It retrieves and processes only the latest data (past 60 seconds) to ensure relevance and uses parallel processing with log rotation to manage system load.

To protect fleet data, the architecture uses layered security: isolated Docker networks, Transport Layer Security (TLS) encryption via an Nginx reverse proxy, and strict routing rules. Only flagged events with minimal metadata (timestamp, location) are sent to the cloud, ensuring privacy and reducing system load.

Figure 3 shows the baseline architecture, where raw data is sent directly to the cloud without edge processing. This model is used to benchmark the improved performance and latency of the proposed FA Architecture.

III. Algorithms for Analysis and Aggregation

This section outlines the core algorithms implemented in our FA architecture, including both the on-board Data Processor Algorithm (See Figure 4) and the off-board Temporal–Spatial ETL Streaming Algorithm (See Figure 5).

The Data Processor Algorithm runs independently on each vehicle to detect abnormal driving behavior and system anomalies in real time. Threshold values are informed by domain-specific heuristics and common telematics practices, and are applied via configurable rule checks to ensure deterministic real-time operation. A background daemon loads configurations—network endpoints, rule checks, and thresholds—and listens for Transmission Control Protocol (TCP) connections. Each connection is handled by a dedicated thread that parses JSON-formatted telemetry data into a queue. A worker thread pool processes this data using rule-based checks to detect events such as: aggressive acceleration (throttle $> 90\%$ for > 15s), harsh braking (brake $> 80\%$ more than 3 times in 5 mins), erratic steering (angle $> 30°$ at least 3 times in 10s), and lane departure risks (yaw error > 0.05 rad or lateral deviation > 0.5 m). It also monitors module health—triggering alerts if any sensor (camera, radar, traffic predictor) reports faults in 10 consecutive readings over 20s—and flags abrupt control transitions (mode changes within 2s). Triggered events are de-duplicated within 5 seconds and encoded into a compact JSON with vehicle ID (i), warning type (r), and event hits (h: timestamps and GPS), enabling efficient, low-bandwidth reporting without raw data transfer.

The Temporal–Spatial ETL Streaming Algorithm runs off-board and aggregates processed warnings in real time. Each event is mapped to a geo-fenced zone based on GPS coordinates, representing regions like delivery routes or test areas. A sliding 15-minute time window, updated every minute, groups events by zone and computes two metrics: the total number of threshold-exceeding events and the number of unique vehicles involved. For example, in brake analysis, it counts high-braking events and identifies how many different trucks triggered them in a given zone and time frame. The output is a JSON object with fields: z (zone ID), ws and we (window start and end in ISO 8601), hb (high brake events), and av (active vehicles). This enables near real-time visibility into fleet behavior, helping managers detect and localize safety concerns by time and location.

IV. Benchmarking and Performance Evaluation

We provide a comprehensive benchmarking and performance evaluation of the proposed FA Architecture, focusing on key latency metrics to assess system responsiveness. The evaluation includes average E2E latency under varying data rates and network conditions. Real-world data is used to compare the baseline and FA architectures, whereas synthetic data is used for controlled experiments. No data filtering is applied during testing. To ensure a detailed analysis, we break down the latency measurements for the FA Architecture into several key components. First, the On-board to Off-board latency refers to

TABLE I.
MODULES AND THEIR ROLES WITHIN THE PROPOSED FEDERATED ANALYTICS (FA) ARCHITECTURE.

Module	**Location & FA Role**	**Key Function & Output Format**
Ingestion Module	On-board (Jetson Orin) ↔ Cloud (supporting)	Collects ROS telemetry and forwards messages to the on-board *Data Processing Module*. Transmits processed data to the cloud; outputs `Telemetry JSON`.
FA Processor (Data Processing Module)	On-board (Jetson Orin) **Core FA module**	Performs real-time anomaly detection with rule-based thresholds; outputs compact `Warning JSON`.
ETL / TSA Streaming	Cloud (post-FA aggregation)	Aggregates warnings in 15-min sliding windows; outputs `Hotspot Summary JSON`.
Storage Service	Cloud	Hierarchical blob storage for long-term archival and retrieval.
Visualization Service	Cloud (interface)	Real-time dashboards and historical analytics for fleet managers.
Inter-Vehicle Data Sharing	On-board (Jetson Orin) ↔ Cloud (FA-enabled sync)	Distributes last 60 s of FA warnings to peer vehicles; outputs `Recent Warnings JSON`.

the delay experienced during the transmission of data from the vehicle to the off-board backend environment. Next, the IoT Service Center to Data Stream ETL latency captures the time taken from when data is enqueued in the IoT Service Center until it is processed by the Data Stream ETL Service. Finally, the Average E2E latency (Vehicle-to-Cloud-to-Vehicle) represents the total time taken for data to travel from the originating vehicle to the cloud, undergo processing, and be transmitted to another connected vehicle. This final metric encompasses all stages of data transmission, processing, and response delivery. We evaluate the system's performance under two distinct experimental setups.

In the first setup, we compare the performance of the proposed Federated Analytics (FA) Architecture with a baseline architecture using real-world data from instrumented Ford F-Max test trucks. In the baseline, raw data is transmitted directly between vehicles via the IoT Service Center without any intermediate analytics. In the FA Architecture, Vehicle 1 performs local analysis on an on-board NVIDIA Jetson Orin device, and only the results are transmitted to the cloud. Vehicle 2 then retrieves the aggregated output from the Storage Service. All tests use unidirectional communication and are repeated five times, with the mean values used for evaluation. The experiment involves three types of data streams: module status (20Hz), Highway Pilot (HP) vehicle status (100Hz), and a combined stream. These are transmitted through the Ingestion Module, which includes a ROS topic consumer for real-time handling. The module status stream reports health indicators, whereas the HP vehicle status stream includes data, such as steering angle, throttle input, and engine signals. Raw message sizes are approximately 180 bytes for module status and 1850 bytes for HP vehicle status. In contrast, the FA Architecture reduces data size by transmitting analytics outputs—generated by the Data Stream ETL Service—as compact JSON messages of up to 200 bytes. These include warnings like steering anomalies and module faults, significantly lowering transmission and storage costs. This reduction is most pronounced for high-frequency streams like HP vehicle status, whereas gains for lower-bandwidth streams like module status are more modest, as event summaries are already compact by nature.

TABLE II.
LATENCY COMPARISON FOR RAW DATA TRANSFER AND FA ARCHITECTURES.

Topics	**Raw (s)**	**FA (s)**
module_status (20 Hz)	0.55	1.03
hpvehiclestatus (100 Hz)	0.62	1.11
Combined Streams	0.746	1.12

Table II compares average end-to-end latency between the raw data transfer architecture and the proposed FA Architecture. For the module_status stream at 20Hz, latency increases from 0.55 seconds (raw) to 1.03 seconds (FA), mainly due to the overhead of retrieving processed data from Blob Storage. A similar pattern is seen with the HP vehicle status stream at 100Hz, where latency rises from 0.62 seconds to 1.11 seconds. For combined streams, the raw architecture yields an average latency of 0.746 seconds, compared to 1.12 seconds under the FA Architecture.

In the second setup, we assess the performance of the proposed FA Architecture under varying data frequencies using synthetic data in a bidirectional communication setup. Both devices process, transmit, and receive information simultaneously. Structured synthetic telemetry messages, each around 3,311 bytes, are generated and sent to test system behavior under different data rates. Each test iteration includes one AUTO and one MANUAL transition message to simulate abrupt control mode changes. To measure end-to-end latency, timestamps are embedded in the JSON analysis results produced by the on-board module, while additional metadata is appended on the backend. The maximum size of analysis results retrieved from blob storage during these tests is 134 bytes.

Table III shows the latency performance of the proposed FA Architecture in a bidirectional setup under varying data frequencies. One instance runs on an NVIDIA Jetson Orin (Vehicle 1), the other on a laptop (Vehicle 2). The measurements

```
Require: Environment variables (hosts, ports, flags, thresholds)
Ensure : Running data-processor service
processor ← DataProcessor(env.vehicle_id)  open TCP socket on
  (listen_host, listen_port)
while true do
    (conn, addr) ← accept()  spawn client_handler(conn, processor)
end
Procedure client_handler(conn, processor):
    foreach line in conn stream do
        row ← parseJSON(line)  processor.enqueue_data(row)
    end
Class DataProcessor(id):
    start log-worker, processing-worker, warning-sender threads
    Procedure enqueue_data(row):
        processing_queue.put(row)
    Procedure processing_worker():
        while true do
            row ← processing_queue.get()  if row is shutdown then
                return
            end
            analyze(row)
        end
    Procedure analyze(row):
        purge_old_data()  update_history(row)  foreach check in
          {throttle, brake, gps, steering, risky, modules} do
            if check.enabled then
                run_check(row)
            end
        end
    Procedure enqueue_warning(text):
        if not duplicate in last 5 s then
            warning_queue.put(text)
        end
    Procedure warning_sender():
        while true do
            w ← warning_queue.get()  if w is shutdown then
                return
            end
            send_over_tcp(w)
        end
```

Figure 4. Data Processor Algorithm.

TABLE III.
LATENCY ANALYSIS UNDER DIFFERENT DATA FREQUENCIES FOR 10-MINUTE DATA FLOW BETWEEN VEHICLES.

Data Freq.	**V1 → V2**			**V2 → V1**		
	On→Off	**IoT→ETL**	**Avg. E2E**	**On→Off**	**IoT→ETL**	**Avg. E2E**
0.5 Hz	0.083s	0.107s	1.18s	0.009s	0.109s	1.07s
1 Hz	0.018s	0.108s	1.05s	0.009s	0.104s	0.87s
5 Hz	0.008s	0.110s	0.76s	0.008s	0.104s	0.76s
10 Hz	0.009s	0.105s	0.80s	0.008s	0.104s	0.79s
20 Hz	0.009s	0.109s	0.86s	0.128s	0.114s	1.10s

include on-board to off-board latency, IoT service to data stream ETL latency, and average E2E latency in both directions. As frequency increases from 0.5Hz to 10Hz, latency values decrease and stabilize, with optimal performance observed between 5Hz and 10Hz. In this range, E2E latency remains between 0.76s and 0.80s in both directions. At 20Hz, latency increases due to higher computational load and queuing. On-board to off-board latency remains stable at 0.009s (Vehicle 1 to 2) and rises to 0.128s (Vehicle 2 to 1). Average E2E latency grows to 0.86s and 1.10s, respectively. IoT service to ETL latency remains steady between 0.104s and 0.110s up to 10Hz,

```
Require: Continuous stream of telemetry events
Ensure : Real-time aggregates pushed to Storage Service
// Main ingestion loop
while event ← ingest_next() do
    event.timestamp ← extract_event_time(event)  if not
      is_anomaly(event) then
        continue
    end
    event.zone ← lookup_geo_fence(event.lat, event.lon)  aggre-
      gate_queue.put(event)
end
Procedure aggregate_worker()
    while true do
        window ← new_window(15 min, 1 min)  buffer ← de-
          queue_events(window)
        foreach zone in buffer.group_by(zone) do
            events_z ← buffer.filter(zone)  hb ← count_if(events_z,
              event.value ¿ threshold)  av ← distinct_count(events_z,
              event.vehicle_id)  result ← { “z”: zone, “ws”: win-
              dow.start, “we”: window.end, “hb”: hb, “av”: av }
              send_to_storage(result)
        end
        sleep_until(next_hop_time())
    end
end
```

Figure 5. Temporal-Spatial ETL Streaming Analysis Algorithm.

with only a slight increase at 20Hz. Furthermore, E2E latency from Vehicle 1 to 2 rises by 13.16% (0.76s to 0.86s), while latency from Vehicle 2 to 1 increases by 44.74% (0.76s to 1.10s). These results highlight the trade-off between data frequency, processing capability, and network performance, stressing the importance of tuning the system for high-frequency workloads.

Beyond latency, the system supports operational effectiveness by producing explainable event warnings and enabling TSA-based hotspot identification for time- and location-correlated insights. Robustness is supported by short-term de-duplication and consecutive fault detection, which help filter transient noise. Finally, scalability is enabled by the event-driven architecture and fixed-window aggregation, which largely decouples cloud processing and uplink costs from raw telemetry volume.

V. Conclusions and Future Work

This paper presents a federated analytics-assisted platform that combines edge computing and cloud services to support real-time monitoring in connected truck fleets. Telemetry data is processed locally on Jetson Orin devices using domain-specific threshold checks to reduce data transmission, preserve privacy, and ensure low-latency responses. Aggregated insights are generated through spatio-temporal analysis in the cloud, and visualized via interactive dashboards for fleet-wide operational awareness. Experimental results show that the system maintains an average E2E latency of approximately 1 second even at 20 Hz data rates, demonstrating scalability and efficiency. Future work will focus on latency optimization through edge-cloud synchronization and 5G integration. By leveraging local processing capabilities, we will also aim to implement lightweight unsupervised learning models that can dynamically adjust thresholds based on individual driver profiles and environmental contexts, further enhancing the robustness of fleet management solutions.

References

[1] 5G Automotive Association, "C-V2X Use Cases, Methodology, Examples and Service Level Requirements," 5G Automotive Association, White Paper, June 2019.

[2] 5G Automotive Association, "Use Case Implementations for Sensor Data Sharing," 5G Automotive Association, White Paper, 2023.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[4] M. C. Lucas-Estañ *et al.*, "V2x service provisioning with 5g v2n2v communications with cross-stakeholder information sharing," in *2024 IEEE Vehicular Networking Conference (VNC)*, 2024, pp. 109–112.

[5] M. C. Lucas-Estañ *et al.*, "An analytical latency model and evaluation of the capacity of 5g nr to support v2x services using v2n2v communications," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 2293–2306, 2023.

[6] L. Zhao *et al.*, "A decentralized communication-efficient federated analytics framework for connected vehicles," *IEEE Transactions on Vehicular Technology*, 2024.

[7] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated anomaly analytics for local model poisoning attack," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 596–610, 2022.

[8] L. Toka *et al.*, "5g on the roads: Latency-optimized federated analytics in the vehicular edge," *IEEE Access*, vol. 11, pp. 81 737–81 752, 2023.

[9] B. Coll-Perales *et al.*, "End-to-end latency of v2n2v communications under different 5g and computing deployments in multi-mno scenarios," in *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2022, pp. 622–627.

[10] B. Coll-Perales *et al.*, "Improving the latency of 5g v2n2v communications in multi-mno scenarios using mec federation," in *2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring)*, 2022, pp. 1–5.

[11] Y. Liu, K. A. Hassan, M. Karlsson, Z. Pang, and S. Gong, "A data-centric internet of things framework based on azure cloud," *IEEE Access*, vol. 7, pp. 53 839–53 858, 2019.

[12] R. Dautov and S. Distefano, "Stream processing on clustered edge devices," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 885–898, 2022.

[13] M. Waly, *Learning Microsoft Azure Storage: Build Large-Scale, Real-World Apps by Effectively Planning, Deploying, and Implementing Azure Storage Solutions*. Packt Publishing Ltd, 2017.