An Edge-Driven Digital Twin Structure for Connected and Autonomous Vehicles

B. Aishwarya
Dept. of CSE (Artificial Intelligence & Machine Learning)
ACE Engineering College *(JNTUH)*
Hyderabad, India
aishwaryabandi147@gmail.com

T. Anirudh Singh
Dept. of CSE (Artificial Intelligence & Machine Learning)
ACE Engineering College *(JNTUH)*
Hyderabad, India
thakuranirudh1920@gmail.com

B.Pradeep
Dept. of CSE (Artificial Intelligence & Machine Learning)
ACE Engineering College *(JNTUH)*
Hyderabad, India
mail2pradeepbandari@gmail.com

Mrs. J. Bhargavi
Assistant Professor, Dept. of CSE (Artificial Intelligence & Machine Learning)
ACE Engineering College *(JNTUH)*
Hyderabad, India
bhargavi.jangam@aceec.ac.in

***Abstract*—Connected and Autonomous Vehicles (CAVs) generate continuous streams of data that must be processed efficiently to support real-time monitoring, safety, and intelligent decision-making. However, handling such high-volume data using centralized systems introduces latency and scalability challenges. This paper presents an edge-based Digital Twin framework designed for real-time vehicular data processing using a time-series architecture. The proposed system integrates a real-world GPS dataset to simulate multiple vehicles and employs MQTT-based communication for lightweight data transmission. A Digital Twin engine deployed at the edge processes incoming telemetry data, performs anomaly detection based on speed variations and behavioral patterns, and stores results in a time-series database for efficient querying. Additionally, a RESTful API layer and interactive map visualization enable real-time monitoring and analysis of vehicle trajectories and detected anomalies. Experimental results demonstrate that the system supports scalable multi-vehicle simulation, efficient data handling, and effective anomaly identification. The proposed framework provides a practical and lightweight solution for intelligent transportation systems and future smart mobility applications.**

***Keywords*—Digital Twin, Edge Computing, MQTT, Time-Series Database, Anomaly Detection, Connected Vehicles, GeoLife Dataset, Real-Time Processing**

I. INTRODUCTION

This template, modified in MS Word 2007 and saved as a "Word 97-2003 Document" for the PC, The rapid evolution of Information and Communication Technologies (ICT) has significantly transformed traditional vehicles into intelligent and connected systems known as Connected and Autonomous Vehicles (CAVs). These vehicles are equipped with advanced sensors, communication modules, and onboard computing systems that continuously generate large volumes of data related to vehicle dynamics, location, and environmental conditions. Efficiently processing and managing this data in real time is essential for enabling applications such as traffic monitoring, predictive maintenance, and intelligent transportation systems.

However, conventional cloud-based approaches for data processing often introduce challenges such as high latency, network congestion, and limited scalability. These limitations become critical in time-sensitive vehicular environments where quick decision-making is required. To address these issues, edge computing has emerged as a promising solution by bringing computation closer to the data source.

In this context, the concept of a Digital Twin (DT) plays a vital role. A Digital Twin is a virtual representation of a physical entity that continuously synchronizes with real-world data to monitor, analyze, and predict system behavior. When deployed at the edge, a Digital Twin enables low-latency data processing, efficient resource utilization, and real-time system insights.

This paper proposes an edge-based Digital Twin framework for connected vehicles that supports real-time data ingestion, processing, and visualization. Unlike traditional approaches that rely on synthetic datasets, the proposed system utilizes a real-world GPS dataset to simulate multiple vehicles and generate realistic mobility patterns. The system integrates lightweight MQTT-based communication for data transmission, a time-series database for efficient storage, and a Digital Twin engine for anomaly detection based on vehicle behavior.

Furthermore, the framework provides a RESTful API for data access and an interactive map-based visualization module to analyze vehicle trajectories and anomalies in real time. The proposed architecture is modular, scalable, and lightweight, making it suitable for practical deployment in smart mobility environments.

The main contributions of this paper are as follows:

- Development of a real-time edge-based Digital Twin framework for vehicular systems
- Integration of real-world dataset replay for realistic simulation
- Design of a hybrid anomaly detection mechanism based on speed and behavioral patterns
- Implementation of a time-series data pipeline using MQTT and InfluxDB
- Visualization of vehicle trajectories and anomalies using interactive mapping tools

II. RELATED WORK

A. Digital Twin (DT)

The concept of Digital Twin (DT) has gained significant attention in recent years due to its ability to create virtual representations of physical systems for real-time monitoring and analysis. Initially developed for manufacturing and aerospace applications, Digital Twin technology is now being widely explored in the domain of Connected and Autonomous Vehicles (CAVs). The integration of DT with edge computing has further enhanced its capabilities by enabling low-latency data processing and real-time decision-making.

Several studies have proposed frameworks for Digital Twin systems in vehicular environments. Existing approaches primarily focus on modeling vehicle behavior, improving traffic efficiency, and enabling predictive maintenance. Many of these systems rely on cloud-based architectures, where data collected from vehicles is transmitted to centralized servers for processing. While such approaches provide scalability, they suffer from high latency and increased network load, making them less suitable for real-time applications.

Recent research has shifted towards edge-based architectures to overcome these limitations. Edge computing allows data processing to occur closer to the source, reducing latency and improving system responsiveness. In this context, Digital Twin frameworks deployed at the edge have been proposed to support applications such as cooperative driving, environmental monitoring, and vehicle diagnostics. These frameworks utilize lightweight communication protocols such as MQTT and standardized data models to ensure efficient and interoperable data exchange.

A notable approach introduces an edge-based Digital Twin framework that integrates vehicle sensing data, communication protocols, and edge computing resources to enable real-time synchronization between physical vehicles and their virtual counterparts. The system demonstrates improved performance in terms of communication efficiency and computational overhead. However, many existing solutions lack practical implementation using real-world datasets and often rely on synthetic data for evaluation.

In addition, several studies have explored anomaly detection techniques in vehicular systems. Traditional methods are primarily rule-based, focusing on predefined thresholds such as speed limits or sensor values. While these approaches are simple and computationally efficient, they often fail to capture dynamic and

unpredictable driving behaviors. More advanced techniques involving machine learning and distributed edge intelligence have also been proposed, but they introduce additional complexity and computational overhead.

Despite these advancements, there remains a gap in developing a lightweight, scalable, and end-to-end Digital Twin framework that integrates real-world data, edge processing, anomaly detection, and visualization in a unified system.

To address these limitations, the proposed work presents a practical edge-based Digital Twin framework that leverages a real GPS dataset for multi-vehicle simulation, employs a time-series database for efficient data handling, and integrates a hybrid anomaly detection mechanism. Unlike existing systems, the proposed framework focuses on simplicity, scalability, and real-time visualization, making it suitable for deployment in smart transportation environments.

III. PROPOSED SYSTEM

This section presents the design and implementation of the proposed edge-based Digital Twin framework for real-time vehicular data processing and anomaly detection. The system is designed as a modular and scalable architecture that enables efficient data ingestion, processing, storage, and visualization using real-world datasets.

A. System Overview

The proposed framework follows a layered architecture where each component performs a specific function in the data processing pipeline. The system integrates real-time data streaming, edge-based processing, and time-series analytics to create a synchronized digital representation of multiple vehicles.

The overall workflow of the system consists of the following steps:

- Real-world GPS data is obtained from the GeoLife dataset
- The dataset is replayed to simulate multiple vehicles
- Data is transmitted using MQTT protocol
- The Digital Twin engine processes incoming data and detects anomalies
- Processed data is stored in a time-series database
- Data is accessed via APIs and visualized on interactive maps

B. System Architecture

The architecture of the proposed system consists of six major components:

1. Data Source Layer

This layer uses the GeoLife GPS dataset to simulate real-world vehicular movement. Unlike synthetic data, this dataset provides realistic mobility patterns, improving the reliability of the system.

2. Data Ingestion Layer

The dataset replay module acts as a virtual vehicle simulator. It reads GPS trajectory files and publishes telemetry data such as vehicle ID, latitude, longitude, and speed to the MQTT broker.

- Example data format:

{

"vehicle_id": "vehicle_01",

"latitude": 39.98,

- "longitude": 116.31,
- "speed": 45

}

3. Communication Layer (MQTT Broker)

The system uses the MQTT protocol for lightweight and efficient data transmission. The publish–subscribe model enables real-time communication between simulated vehicles and the Digital Twin engine.

4. Digital Twin Processing Layer

This is the core component of the system. The Digital Twin engine performs the following functions:

- Subscribes to incoming telemetry data
- Processes and structures the data
- Performs anomaly detection
- Stores processed data in the database

The Digital Twin maintains a real-time virtual representation of each vehicle, enabling continuous monitoring and analysis.

5. Storage Layer (Time-Series Database)

The system uses InfluxDB to store both telemetry and anomaly data. Since vehicle data is time-dependent and continuously generated, a time-series database is more efficient than traditional relational databases.

Two main measurements are used:

- **vehicle_telemetry**: Stores speed, location, and timestamp
- **vehicle_anomaly**: Stores anomaly details such as type and location

6. Application and Visualization Layer

This layer provides user interaction and data interpretation:

- A Flask-based REST API enables data access
- Visualization modules generate interactive maps using Folium
- Vehicle paths are displayed as trajectories
- Anomalies are highlighted as markers

C. *Anomaly Detection Mechanism*

The system implements a hybrid anomaly detection approach combining rule-based logic and probabilistic control to simulate realistic vehicle behavior.

The following conditions are used for anomaly detection:

- **Overspeed Detection**: Triggered when vehicle speed exceeds a predefined threshold
- **Sudden Speed Change**: Detected when the difference between consecutive speed values exceeds a limit
- **Probabilistic Triggering**: Introduced to simulate real-world unpredictability

To prevent excessive anomaly generation, the system includes control mechanisms:

- **Cooldown Period** to limit frequency
- **Maximum anomaly** count per vehicle
- **State memory (last_speed)** for temporal analysis

This approach ensures that anomalies are meaningful, controlled, and realistic.

D. *Data Flow and Processing Pipeline*

The data flow in the system is continuous and real-time:

- Vehicles (simulated) generate telemetry data
- Data is published to MQTT topics
- The Digital Twin engine consumes and processes the data
- Processed data is stored in InfluxDB
- APIs and visualization tools retrieve data for analysis

This pipeline enables efficient handling of high-frequency data streams while maintaining low latency.

IV. IMPLEMENTATION AND EXPERIMENTAL SETUP

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

A. *Development Environment*

The system is implemented using a combination of lightweight and widely adopted technologies to ensure scalability and ease of deployment. The development is carried out in a Linux-based environment using Windows Subsystem for Linux (WSL).

The primary technologies used are as follows:

- **Programming Language**: Python
- **Messaging Protocol**: MQTT (Eclipse Mosquitto)
- **Database**: InfluxDB (Time-Series Database)
- **Backend Framework**: Flask (REST API)
- **Visualization Tools**: Folium and Leaflet
- **Containerization**: Docker

These technologies enable efficient data processing, communication, and visualization in a modular architecture.

B. *System Components*

The implementation consists of multiple interconnected modules, each responsible for a specific function:

1) Dataset Replay Module

The dataset replay module simulates real vehicles using the GeoLife GPS dataset. It reads trajectory files and publishes real-time telemetry data to the MQTT broker at fixed intervals. The module also computes vehicle speed dynamically using geographical distance calculations.

2) MQTT Communication Module

The MQTT broker acts as the communication backbone of the system. It facilitates real-time data exchange between the dataset replay module and the Digital Twin engine using a publish–subscribe mechanism.

3) Digital Twin Engine

The Digital Twin engine is implemented as a Python-based processing module. It subscribes to telemetry data, processes incoming messages, and performs anomaly detection. The engine also manages state information such as previous speed values and anomaly control parameters.

4) Database Module

InfluxDB is used to store both telemetry and anomaly data. It is optimized for handling high-frequency time-series data and allows efficient querying over time intervals. Data is stored in structured measurements for easy retrieval and analysis.

5) API Module

A Flask-based REST API is developed to provide access to stored data. It enables users to query the latest vehicle information and retrieve historical data for analysis.

6) Visualization Module

The visualization component generates interactive maps using Folium. Vehicle trajectories are displayed as colored paths, while detected anomalies are marked using visual indicators. This provides an intuitive way to analyze vehicle behavior.

C. Visualization and Analysis

The visualization module provides an intuitive representation of vehicle behavior through interactive maps. Vehicle trajectories are displayed as continuous paths, while detected anomalies are marked using distinct visual indicators.

As shown in the output results (refer to visualization figures in the implementation section), the system clearly highlights anomaly locations along vehicle paths. This allows users to easily identify patterns such as frequent
speed variations or abnormal driving segments.

vi. Conclusion and Future Work

This paper presented an edge-based Digital Twin framework for real-time vehicular data processing and anomaly detection using a time-series architecture. The proposed system integrates real-world GPS data, lightweight MQTT-based communication, and efficient storage using a time-series database to create a scalable and modular solution for connected vehicle environments.

The Digital Twin engine successfully processes continuous data streams, detects anomalies based on vehicle behavior, and maintains synchronization between simulated vehicles and their virtual counterparts. The use of a real-world dataset enhances the realism and reliability of the system, while the visualization module provides an intuitive interface for analyzing vehicle trajectories and detected anomalies.

Experimental results demonstrate that the proposed framework achieves efficient real-time processing, low communication overhead, and scalable multi-vehicle simulation. The system effectively combines simplicity and performance, making it suitable for practical deployment in intelligent transportation systems.

However, the current anomaly detection mechanism is primarily based on rule-based logic and probabilistic control. As part of future work, the system can be extended by integrating advanced machine learning and deep learning models to improve detection accuracy and adaptability. Additionally, incorporating real-time data from physical vehicles and IoT sensors can further enhance the system's applicability in real-world scenarios.

Future enhancements may also include distributed edge-cloud architectures, improved security mechanisms, and large-scale deployment for smart city applications.

REFERENCES

[1] [1] C. Campolo, G. Genovese, A. Molinaro, B. Pizzimenti, G. Ruggeri, and D. M. Zappalà, "An edge-based digital twin framework for connected and autonomous vehicles: Design and evaluation," IEEE Access, vol. 12, pp. 46290–46300, 2024.

[2]

[3] [2] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," IEEE Access, vol. 7, pp. 167653–167671, 2019.

[4]

[5] [3] Z. Wang, R. Gupta, K. Han, H. Wang, A. Ganlath, N. Ammar, and P. Tiwari, "Mobility digital twin: Concept, architecture, case study, and future challenges," IEEE Internet of Things Journal, vol. 9, no. 18, pp. 17452–17467, 2022.

[6]

[7] [4] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of Vehicles: Challenges and opportunities," Proceedings of the IEEE, vol. 108, no. 2, pp. 308–323, 2020.

[8]

[9] [5] S. Lu, N. Ammar, A. Ganlath, H. Wang, and W. Shi, "A comparison of end-to-end architectures for connected vehicles," in Proc. IEEE MetroCAD, 2022, pp. 72–80.

[10]

[11] [6] Eclipse Foundation, "Mosquitto MQTT Broker," [Online]. Available: https://mosquitto.org/

[12]

[13] [7] InfluxData, "InfluxDB: Open Source Time Series Database," [Online]. Available: https://www.influxdata.com/

[14]

[15] [8] Flask Documentation, "Flask Web Framework," [Online]. Available: https://flask.palletsprojects.com/

[16]

[17] [9] Microsoft Research, "GeoLife GPS Trajectories Dataset," [Online]. Available: https://www.microsoft.com/en-us/research/project/geolife-building-social-networks-using-human-location-history/

[18]

[19] [10] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2016. (optional if you mention ML future work)

[20]

[21] [11] Y. Liu et al., "DeepEdge: A new deep learning framework for edge computing," IEEE Access, 2019.

[22]

[12] X. Zeng et al., "Edge SGD: Efficient distributed learning at the edge," IEEE Transactions on Networking, 2021.