

EDGE – FOG CONTROLLED AUTONOMOUS VEHICLE COMMUNICATION

M Sarath Chandra¹, A Sampreethi², Md Naziya³, Ch Abhinaya⁴

¹Dept. of Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India & Address

²Dept. of Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India & Address

³Dept. of Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India & Address

⁴Dept. of Electronics and Communication Engineering, MVSR Engineering College, Hyderabad, India & Address

Abstract -This project presents a hybrid Edge-Fog computing framework for autonomous vehicle communication within an intelligent transportation environment. The system focuses on minimizing decision latency and enhancing road safety by distributing computational tasks across edge nodes and fog infrastructure instead of relying solely on centralized cloud systems. A simulated multi-lane highway scenario is developed where vehicles dynamically interact using Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure(V2I) communication. Python is used to model traffic behaviour and visualization, while MATLAB handles real-time decision-making, including lane selection, speed control, and collision avoidance. The system incorporates adaptive offloading strategies based on network conditions, vehicle state, and infrastructure load. Experimental results demonstrate improved responsiveness, reduced latency, and efficient traffic coordination, indicating the effectiveness of integrating edge and fog computing in future autonomous transportation systems.

Key Words: Edge computing, Fog computing, Task Offloading, Traffic Simulation, Collision Avoidance, Lane Change Optimization, Road Side Units

1. INTRODUCTION

Autonomous driving systems depend heavily on continuous data exchange and fast decision making. Vehicles must react instantly to surrounding conditions such as traffic density, obstacles, and speed variations. Any delay in processing can lead to unsafe situations.

Traditional centralized computing models rely on remote servers, which introduce communication delays due to long transmission distances. These delays become critical in applications where decisions must be made within milliseconds.

To address this limitation, distributed computing approaches have been introduced. Edge computing enables data processing near the source, while fog computing provides an intermediate layer that aggregates regional information.

Together, these technologies create a hierarchical processing structure that balances speed and computational capability. This project focuses on designing a system where vehicles operate collaboratively using decentralized

communication and intelligent decision mechanisms, ensuring safe and efficient traffic flow.

2. LITERATURE SURVEY

Existing research in autonomous transportation primarily explores cloud-based framework for vehicle coordination. While these systems offer high computational power, they suffer from significant communication delays, making them unsuitable for real-time decision-making scenarios. Studies have highlighted that latency beyond a few milliseconds can lead to unsafe driving conditions.

Recent advancements have shifted focus toward edge computing, where processing is performed near data sources. Research demonstrates that edge-based architectures reduce response time and improve system reliability. However, edge nodes alone may lack sufficient computational resources for complex analytics, leading to the integration of fog computing as a supplementary layer.

IoV based communication models have also gained attention for enabling cooperative driving. V2V communication facilitates direct interaction between vehicles, improving awareness of nearby traffic conditions, while V2I communication connects vehicles to infrastructure for broader environmental insights. Despite these advancements, challenges remain in efficiently distributing computational tasks and maintaining network stability under dynamic conditions.

This project builds upon these concepts by combining edge and fog computing with IoV communication into a unified framework, addressing both latency and scalability challenges.

3. METHODOLOGY

The proposed system follows a structured simulation based methodology designed to emulate real-world autonomous vehicle coordination under dynamic traffic conditions. The approach integrates vehicle behaviour modelling, communication mechanisms, and distributed computing decisions into a continuous execution cycle.

The process begins with environment initialization, where a multi-lane highway is defined with fixed spatial dimensions and directional constraints. Vehicles are introduced into this environment with randomized attributes such as position, velocity, and lane assignment. This randomness is not

arbitrary—it is essential to prevent predictable patterns and to replicate heterogeneous traffic conditions observed in real scenarios.

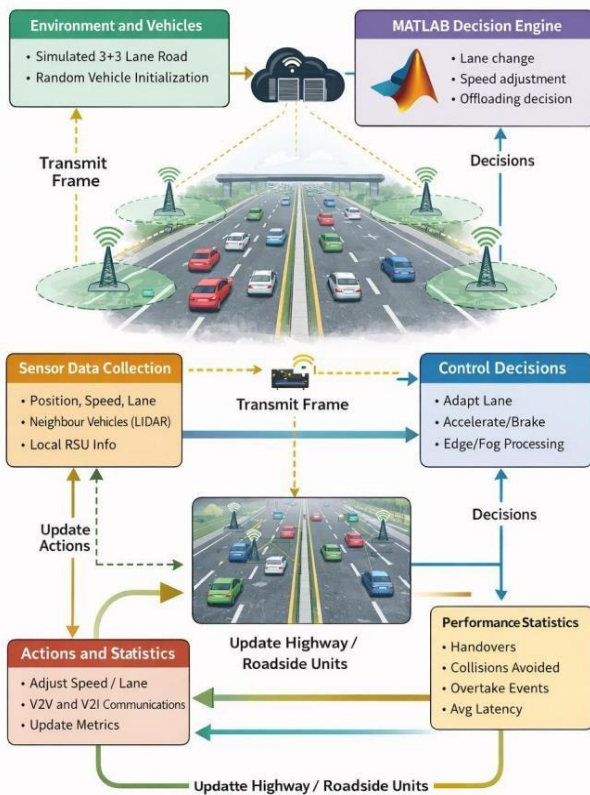


Fig-1 Edge – Fog/IOV Dynamic Control for 3+3 lane Highway

At each simulation interval, vehicle states are updated using discrete-time motion equations. Instead of continuous modelling, a step-based update (with fixed time intervals) is used to balance computational efficiency and simulation accuracy. This allows the system to scale while still maintaining realistic motion behavior. Boundary conditions are handled using a wrap-around logic, ensuring uninterrupted vehicle flow and eliminating artificial stopping points. A key part of the methodology lies in situational awareness generation. Each vehicle continuously scans its surroundings to identify nearby entities within a predefined communication radius. This local perception is constructed using relative position, lane alignment, and speed differences. Rather than relying on global knowledge, the system intentionally limits awareness to nearby vehicles, mimicking real-world sensor constraints and decentralized intelligence. The collected environmental data is then transformed into a structured representation known as a sensor frame. This step is critical because raw simulation data is not directly usable for decision-making. The frame consolidates spatial, dynamic, and network-related parameters into a unified format, enabling efficient

processing by the decision engine. Care is taken to include both physical metrics (like speed and position) and network indicators (such as signal strength and infrastructure load), ensuring that decisions consider both traffic and communication conditions.

Once the data is prepared, it is transmitted to the decision-making module, where computational intelligence is applied. Instead of embedding logic directly into the simulation, the system separates decision processing into an external controller. This separation is intentional it improves modularity, allows independent tuning of algorithms, and reflects real-world deployment where vehicles rely on external computational support. The decision engine evaluates multiple factors simultaneously. It prioritizes safety by analyzing inter-vehicle distances and predicting potential conflicts. Lane selection is treated as a constrained optimization problem, where only feasible neighboring lanes are considered based on direction and available space. Speed adjustments are derived from proximity conditions rather than fixed rules, allowing adaptive responses to traffic density.

Another critical component is the computation offloading strategy. The methodology does not assume that all decisions should be processed locally or remotely. Instead, it dynamically determines the execution location based on system conditions. Factors such as vehicle speed, surrounding traffic density, communication quality, and infrastructure load are evaluated before making an offloading decision. This ensures that the system avoids unnecessary communication delays while still leveraging external computational resources when beneficial. After decisions are generated, they are reintegrated into the simulation environment. Vehicles update their states accordingly, modifying their trajectory, velocity, and processing mode. This closed-loop interaction between simulation and decision engine creates a feedback system where each cycle influences the next, enabling emergent behavior rather than scripted motion.

Finally, the methodology incorporates continuous monitoring of system performance. Metrics such as latency, safety events, and communication transitions are recorded during execution. This is not just for reporting it allows validation of whether the design choices (like edge offloading or communication ranges) actually improve system behavior under varying conditions.

4.SYSTEM ARCHITECTURE

The system follows a hierarchical distributed architecture designed to minimize delay while maintaining coordinated decision-making across multiple vehicles. Instead of relying on a single centralized unit, computation is divided across different

layers based on urgency and complexity. At the lowest level, vehicles act as intelligent agents that continuously sense their surroundings and exchange information with nearby entities. They do not operate blindly—each vehicle builds a localized understanding of traffic using short-range communication and onboard sensing. This ensures immediate awareness without waiting for external processing.

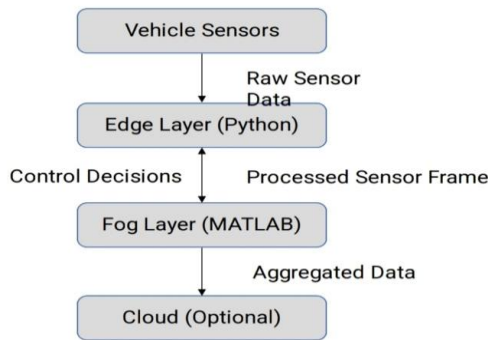


Fig-2 System Architecture of the proposed system

The intermediate layer consists of Road Side Units, which function as edge nodes. These units handle time-sensitive computational tasks that are too heavy for individual vehicles but still require low latency. Their placement along the road ensures that vehicles can quickly connect to the nearest unit, reducing communication delay and enabling faster response.

Above this, the fog layer serves as a regional coordinator. It does not deal with individual vehicle actions directly but instead processes aggregated data to support broader traffic-level insights. This separation prevents overload at lower layers and improves scalability when vehicle density increases.

The decision-making layer is implemented through an external controller that evaluates incoming data and generates optimized actions. By isolating decision logic from simulation, the system achieves flexibility, allowing algorithm improvements without modifying the entire structure.

Overall, the architecture is designed around proximity-based processing: immediate decisions occur close to the vehicle, while complex analysis is handled at higher layers. This distribution is what reduces latency and avoids bottlenecks

5.IMPLEMENTATION

The system is implemented as a co-simulation framework combining Python for environment modelling and MATLAB for decision computation. The core idea is to separate simulation logic from control

intelligence while maintaining real-time interaction between them. On the Python side, the road environment and vehicles are modelled using object-oriented design. Each vehicle instance maintains its own dynamic state and updates it at every simulation step based on motion equations. A scheduler controls the execution loop, ensuring consistent time progression and synchronized updates across all entities. To enable interaction between vehicles, a proximity-based search mechanism is implemented. Instead of checking every possible pair inefficiently, distance filtering is applied to identify only relevant neighboring vehicles within a defined range. This reduces computational overhead and keeps the simulation scalable as the number of vehicles increases. A structured data interface is then created to bridge Python and MATLAB. Instead of passing raw variables, all relevant parameters are packaged into a standardized data structure. This ensures compatibility between the two environments and avoids data inconsistency during transmission. MATLAB component is responsible for executing the decision-making algorithms that govern vehicle behavior. Upon receiving structured data from the Python simulation, MATLAB processes the inputs using control logic that may include rule-based systems or optimization techniques to determine appropriate actions such as speed adjustment, lane changes, or collision avoidance. The computed decisions are then transmitted back to the Python environment, where they are applied to update each vehicle’s state in the next simulation cycle.

```

% 1) Find front vehicle in current lane (before control)
front_dx_before = inf;
front_speed_before = speed;

if ~isempty(lidar)
    for i = 1:size(lidar,1)
        px = lidar(i,1);
        py = lidar(i,2);
        pz = lidar(i,3);

        if plan == lane_idx
            continue;
        end

        dx = px - x;
        if dx > 0 && dx < front_dx_before
            front_dx_before = dx;
            front_speed_before = pz;
        end
    end

    collision_risk_before = (front_dx_before < collision_thresh);

% 2) Decide lane (overtake / stay) using IOW information
target_lane = lane_idx;

% Motivation to overtake: front car close & significantly slower
need_overtake = collision_risk_before || ...
    (front_dx_before < overtake_trigger && ...
    front_speed_before < speed - dv_overtake);

% Candidate lanes within the same direction group
% (no wrong side crossing)
candidate_lanes = [];
if lane_idx == 1 && group_high
    candidate_lanes(end+1) = lane_idx + 1;
end
if lane_idx == 1 && group_low
    candidate_lanes(end+1) = lane_idx - 1;
end

if need_overtake
    for k = 1:length(candidate_lanes)
        l = candidate_lanes(k);
        safe_lane = true;

        if ~isempty(lidar)
            for i = 1:size(lidar,1)
                px = lidar(i,1);
                py = lidar(i,2);
                if plan == l
                    continue;
                end

                dx = px - x;
                % Any vehicle between -gap_behind and +gap_ahead is unsafe
                if dx > -gap_behind && dx < gap_ahead
                    safe_lane = false;
                    break;
                end
            end
        end
        if safe_lane
            target_lane = l;
            break; % Choose the first safe candidate lane
        end
    end
end

% 3) Collision status after control
collision_risk_after = (front_dx_after < collision_thresh);
collision_risk_avoided = 1 - collision_risk_after;

% Collision avoided if there was a risk before and no risk after
% Collision_risk_avoided = 1;

% Speed decision:
if front_dx_after < collision_thresh
    % Immediate danger -> strong brake
    resp.accel = -1;
elseif front_dx_after < safe_headway && target_lane == lane_idx
    % Too close, could not overtake -> brake
    resp.accel = -1;
elseif speed < v_des && speed < v_max
    % Free ahead or successful overtaking -> speed up to desired
    resp.accel = 1;
else
    resp.accel = 0;
end

% 4) Edge/Fog offloading decision (IOV)
overspeed = speed > v_des;
congested = local_density > critical_density;
good_link = (rsu_load < max_rsu_util * rsu_capacity);
rsu_ok = (rsu_load < max_rsu_util * rsu_capacity);

% Offload if overspeed or congestion and RSU/Fog are suitable
if (overspeed || congested) && good_link && rsu_ok
    resp.offload = 1;
else
    resp.offload = 0;
end
    
```

Fig-3 Program for Over Speed of the Vehicle

Communication between Python and MATLAB is achieved using the MATLAB Engine API. Python invokes the MATLAB function, transfers the structured data, and waits for the response. This synchronous interaction guarantees that each simulation step uses updated decisions before proceeding further. Within MATLAB, the control logic is implemented as a modular function. The function evaluates incoming data using conditional rules and generates outputs that directly influence vehicle behavior. The logic is designed to be lightweight so that it can process multiple vehicles efficiently without introducing delays. On receiving the controller output, Python applies the decisions immediately by updating vehicle attributes such as lane position, velocity, and processing mode. Instead of abrupt changes, constraints are applied to ensure transitions remain realistic and stable.

The implementation also includes a resource management mechanism for RSUs. Each unit tracks its current processing load, and updates are applied dynamically as vehicles offload tasks. This prevents unrealistic unlimited processing and introduces practical system limitations. To maintain continuity in communication, a handover mechanism is implemented. When a vehicle moves beyond the coverage of one unit, it automatically switches to another based on proximity. This transition is handled seamlessly without interrupting the simulation flow.

Finally, a visualization module renders the entire system in real time. It reflects vehicle movement, communication links, and processing states, allowing direct observation of system behaviour. Alongside this, performance data is continuously logged for evaluation.

6.RESULTS AND DISCUSSION

The simulation demonstrates that the integration of edge and fog computing significantly enhances system performance compared to traditional centralized approaches. Vehicles exhibit improved responsiveness due to reduced processing delays, enabling timely decision-making in critical scenarios

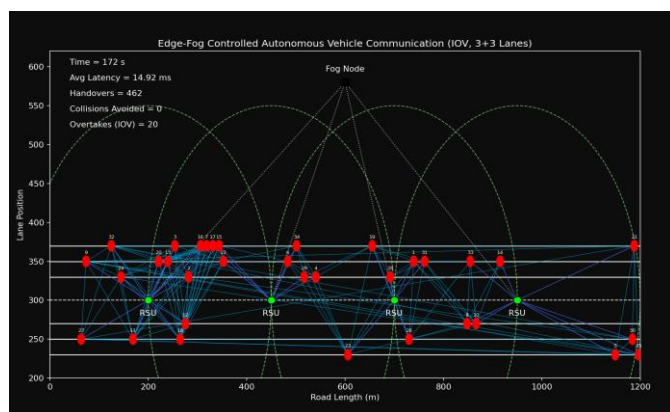


Fig-3 Real time simulation Result

Collision avoidance mechanisms effectively maintain safe distances between vehicles, reducing the likelihood of accidents. The implementation of intelligent overtaking strategies contributes to smoother traffic flow and minimizes congestion

Task offloading to edge nodes is observed to balance computational load efficiently, preventing overload in any single component. Additionally, the system adapts dynamically to varying network conditions, ensuring consistent performance. Latency measurements indicate a notable reduction when edge processing is utilized, validating the effectiveness of the proposed architecture. Overall, the results highlight the importance of distributed computing in achieving reliable and efficient autonomous vehicle systems.

7. CONCLUSION

This work presents a comprehensive approach to autonomous vehicle communication using a hybrid Edge-Fog architecture. By combining decentralized computing with IoV communication models, the system addresses key challenges related to latency, scalability, and safety. The integration of MATLAB-based decision-making with Python simulation enables a flexible and powerful framework for analyzing traffic behavior. The results confirm that distributing computational tasks across multiple layers improves system efficiency and enhances real-time responsiveness.

While the current implementation provides a strong foundation, further enhancements such as machine learning-based decision models and advanced network simulations can extend its capabilities. The proposed system represents a significant step toward the realization of intelligent and adaptive transportation systems for future smart cities.

REFERENCES

- [1] A. Thakur, R. Malekian, Fog computing for detecting vehicular congestion, an internet of vehicles based approach: a review. *IEEE Intell. Transp. Syst. Mag.* 11(2), 8–16 (2019)
- [2] R. Naqvi, S. Salman, S. Wang, M. Ahmed, M. Anwar, A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. *Wirel. Commun. Mob. Comput.* 2019, 1–19 (2019). <https://doi.org/10.1155/2019/3159762>
- [3] R. Mahmud, A.N. Toosi, K. Rao, R. Buyya, Context-aware placement of Industry 4.0 applications in fog computing environments. *IEEE Trans. Ind. Inf.* 16(11), 7004–7013 (2020).
- [4] R.S. Sandhya Devi, V.R. Vijaykumar, P. Sivakumar, Neeraja Lakshmi A, Vinoth Kumar B, in *Edge Architecture Integration of Technologies: Cases on Edge Computing and Analytics* (2020)
- [5] R. Mahmud, K. Ramamohana Rao R. Buyya, *Application Management in Fog Computing Environments: A*

Taxonomy, Review and Future Directions (University of Melbourne, Melbourne, 2020)

- [6] R. Mahmud, K. Ramamohana Rao R. Buyya, Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions (University of Melbourne, Melbourne, 2020)
- [7] H.A. El Zouka, A secure interactive architecture for vehicular cloud environment, in proceedings of IEEE Conference on Smart Cloud (2016), pp. 254-261
- [8] M. Anwar, S. Wang, M. Zia, A. Jadoon, U. Akram, R. Naqvi, S. Salman, Fog computing: an overview of big IoT data analytics. *Wireless. Commun. Mob. Compute.* 2018, 1-22 (2018). <https://doi.org/10.1155/2018/7157192>
- [9] M. Anwar, S. Wang, M. Zia, A. Jadoon, U. Akram, R. Naqvi, J. Sun, Q. Gu, T. Zheng, P. Dong, Y. Qin, Joint Communication and computing resource allocation in vehicular edge computing. *Int. J. Distrib. Sens. Netw.* 15, 155014771983785 (2019).