









# Distributed Edge Computing System for Vehicle Communication

Rinith Pakala<sup>1</sup><sup>a</sup>, Niket Kathiriya<sup>1</sup><sup>b</sup>, Hossein Haeri<sup>2</sup><sup>c</sup>, Satya Prasad Maddipatla<sup>3</sup><sup>d</sup>,  
Kshitij Jerath<sup>2</sup><sup>e</sup>, Craig Beal<sup>4</sup><sup>f</sup>, Sean Brennan<sup>3</sup><sup>g</sup> and Cindy Chen<sup>1</sup><sup>h</sup>

<sup>1</sup>Computer Science Department, University of Massachusetts Lowell, 220 Pawtucket St, Lowell, U.S.A.

<sup>2</sup>Mechanical Engineering Department, University of Massachusetts Lowell, Lowell, U.S.A.

<sup>3</sup>Mechanical Engineering Department, The Pennsylvania State University, University Park, U.S.A.

<sup>4</sup>Mechanical Engineering Department, Bucknell University, Lewisburg, U.S.A.

**Keywords:** Edge Computing, Middleware, Data Repository Server, Safety Critical Transmission.


**Abstract:** The development of communication technologies in edge computing has fostered progress across various applications, particularly those involving vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. Enhanced infrastructure has improved data transmission network availability, promoting better connectivity and data collection from IoT devices. A notable IoT application is with the Intelligent Transportation System (ITS). IoT technology integration enables ITS to access a variety of data sources, including those pertaining to weather and road conditions. Real-time data on factors like temperature, humidity, precipitation, and friction contribute to improved decision-making models. Traditionally, these models are trained at the cloud level, which can lead to communication and computational delays. However, substantial advancements in cloud-to-edge computing have decreased communication relays and increased computational distribution, resulting in faster response times. Despite these benefits, the developments still largely depend on central cloud sources for computation due to restrictions in computational and storage capacity at the edge. This reliance leads to duplicated data transfers between edge servers and cloud application servers. Additionally, edge computing is further complicated by data models predominantly based on data heuristics. In this paper, we propose a system that streamlines edge computing by allowing computation at the edge, thus reducing latency in responding to requests across distributed networks. Our system is also designed to facilitate quick updates of predictions, ensuring vehicles receive more pertinent safety-critical model predictions. We will demonstrate the construction of our system for V2V and V2I applications, incorporating cloud-ware, middle-ware, and vehicle-ware levels.


## 1 INTRODUCTION


Edge computing is an approach that involves situating computational resources near the device in use, allowing for real-time updates and expedited decision-making (ISO/IEC, 2020),(Satyanarayanan et al., 2009). This method leverages edge network resources alongside cloud networks to facilitate in-


formation exchange between local edge devices and global data trends at the cloud. When managing vast amounts of data, efficient distribution between edge and cloud networks is crucial for the transmission and processing of that data.


In Vehicle-to-Infrastructure (V2I) networks, a majority of vehicles can exchange crucial safety information with Road Side Units (RSU), which are infrastructure devices situated along the road (Barrachina et al., 2013). These RSUs, as components of the edge infrastructure, transmit data to cloud servers for centralized computing and cache the results at the edge network to provide response to edge devices. This process results in redundant data transfers, increased computational complexity, communication relays, and the under-utilization of edge resources


<sup>a</sup> <https://orcid.org/0000-0001-7442-4170>


<sup>b</sup> <https://orcid.org/0000-0002-4146-3402>


<sup>c</sup> <https://orcid.org/0000-0002-6772-6266>

<sup>d</sup> <https://orcid.org/0000-0002-5785-3579>

<sup>e</sup> <https://orcid.org/0000-0001-6356-9438>

<sup>f</sup> <https://orcid.org/0000-0001-7193-9347>

<sup>g</sup> <https://orcid.org/0000-0001-9844-6948>

<sup>h</sup> <https://orcid.org/0000-0002-8712-8108>

which are often only used for replicating results or conducting partial computations. Additionally, these devices may have limited deployment along roads, leading to gaps in network coverage. These factors highlight the necessity for an improved architectural setup in edge computing that can seamlessly connect with cloud networks.

Connected autonomous vehicles would obtain more pertinent information regarding road conditions (Tang et al., 2021), enhancing decision-making for speed planning. IoT devices, such as security cameras, would perform more effectively by conducting real-time video analytics at the edge (Mendki, 2018) rather than relying on communication with a central repository.

Our approach to implementing this architecture aims to address two specific challenges: i) reducing response time when handling edge device requests, and ii) providing higher throughput of request rates compared to a central cloud repository. The performance evaluation benchmarks taken into account include assessing the influence of longer road lengths on the variability of response times and examining the consequences of a higher number of vehicle requests on the response rates delivered.

The paper is organized as follows: Section 2 discusses existing architectural designs and the concept of data reduction, addressing related work in current architecture implementation. Section 3 delves into the architectural components and the associated setup. Section 4 presents experimental performance results and analysis of the work. Finally, Section 5 concludes the paper and outlines directions for future work.

## 2 RELATED WORK

### 2.1 IoT Cloud Computing

Autonomous vehicles, which facilitate driving and make decisions based on road conditions, heavily rely on sensor data and are equipped with numerous sensors that generate vast amounts of data (Van Rijmenam, 2017). This locally generated data may be corrupted due to faulty sensors or insufficient road area coverage, necessitating estimation in conjunction with data from other devices to accurately assess road conditions. When processing this data at a central server, both the data volume and communication distance significantly impact the response time. The transfer of substantial amounts of data leads to considerable delays, even with minor increases in distance.

In the current cloud server processing setup, the

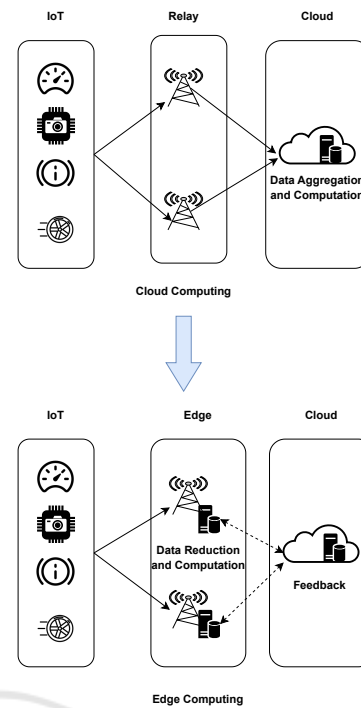


Figure 1: Cloud and Edge processing.

large volume of data stored in the server results in delays in data insertion, processing, and response provision. Real-time updates of road conditions are necessary to deliver accurate information to IoT devices, but outdated information may be received due to these delays. Overcoming such delays can be achieved by employing edge server computing, which gathers information for its corresponding range of IoT devices (Wang et al., 2016).

### 2.2 Edge Limitation

When handling large amounts of data, the edge servers need to handle excessive throughput and rigorous computation. When processing information like friction estimates, which depend hugely on historic data (Panahandeh et al., 2017), the storage and computational limitation of the edge server restrict the prediction ability to provide estimates. Along with data processing, a simple SELECT query to fetch the relevant information is delayed. This delay becomes more pronounced when combined with increased requests from IoT devices.

With the existing edge servers, the mobile edge servers provide caching and transfer required information between the servers (Li et al., 2018). This load-balancing data transfer among edge servers causes increased data duplicity of stored data and increased amount of data transfers between edge

servers. Works are also performed to reduce node communication by using multi join query processing (Kurunji et al., 2013) with all the available data.

We develop our architecture to incorporate data reduction, aiming to decrease the data required for generating timely friction estimates (figure 1). The distributed processing network of edge servers reduces the amount of data processed at each edge server while increasing the availability of servers to handle requests from edge devices.

### 2.3 Allan Variance

Friction estimates rely on both current and historical friction data supplied by vehicles. The computation of friction estimation is directly proportional to the amount of data used during processing, resulting in increased computational latency as data storage grows. This computation can be simplified by determining the relevance of the data in providing friction estimates at the present time.

We build on our previous work on the granulation of large temporal databases (Sinanaj et al., 2022), which employed the Allan variance technique (Allan, 1966) to assess the variation level in friction data. This approach determined the optimal window length of similarity for segmenting the data into batches, generating friction estimates while preserving the overall friction variation distribution (Figure 2). This method proved to be less time-consuming than heuristic-based clustering algorithms (Kodinariya and Makwana, 2013). As the influx of new data from edge devices grows, the friction variation over time cannot be estimated using a constant window length. Our previous work on optimal moving average estimation (Haeri et al., 2022) addresses this issue by proposing a dynamic Allan variance approach to determine the optimal window length. We employ the R\* tree data representation (Beckmann et al., 1990) to extend this work in a multidimensional space involving spatial and temporal attributes. The resulting aggregation window length is used to select the level of the node in the R\* tree. This technique enables the granulation of all records referenced by a tree node, thereby generating hyper-rectangles known as granules.

Each granule encompasses a friction attribute, which is calculated as the average of all data points contained within it. Over time, these granules are re-generated as a function of the recently received edge device data and previously existing granules, forming new granules. The most recent granules in time are utilized to provide friction estimation responses to the edge devices.

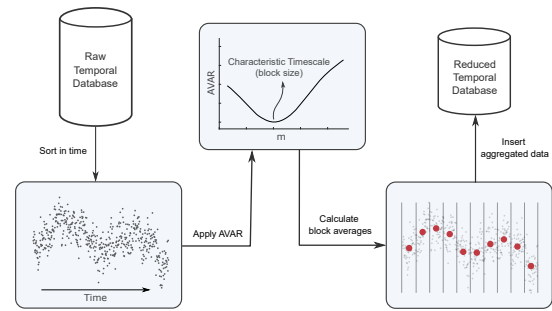


Figure 2: Data granulation using Allan variance. Adapted from (Sinanaj et al., 2022).

## 3 SYSTEM ARCHITECTURE

This section presents the components and architecture setup for a cloud-edge coordinated system. The architectural design is illustrated using vehicle communication with edge and cloud servers to query necessary road path information. The edge servers serve as an immediate source for providing vehicles with required road information and are connected to the cloud for maintaining data relevancy, as well as handling failures at the middleware level. All communication between edge devices, edge servers, and the cloud is assumed to occur over internet protocol (IP).

### 3.1 Edge Device

Edge devices are remote systems with less computational power and depend on the data insights that are provided to them to help with their course of action. In turn, these edge devices provide the required data that it collects to edge and cloud servers (Velmurugadass et al., 2021). In applications such as vehicular systems, the edge device is a vehicle traveling on a roadway. For safety-critical planning of speed and trajectory, the vehicle needs information pertaining to road friction on the upcoming roadway. This information may be obtained from edge server infrastructure positioned on the roadside. Based on the work proposed by (Gao et al., 2022) the minimum preview distance for the vehicle is determined and the edge device generates requests with location points, which are responded to by the edge server with the friction estimates, enabling safe vehicle operation. In turn, vehicles transmit the friction experienced on the road and are used to update the estimates.

### 3.2 Middleware

Middleware are the edge servers placed nearby the edge devices. These are the roadside unit computer

that can send and transmit data over radio (Ou et al., 2019) or wireless networks(Cai and Lin, 2008) The transmitting unit of the middleware has high bandwidth and uses Dedicated Short Range Communication (DSRC). Middleware contains a local database and are equipped with processing power. These middleware servers when placed alongside the highway or at traffic signaling, are useful to exchange friction information with the vehicles traveling on road. The middleware performs computations that combine historical data with the latest friction data collected from vehicles. The revised roadway friction estimates can then be queried by the same or other vehicles on the roadway.

### 3.3 Cloud Server Repository

Cloud server acts as a data repository for the data collected by edge servers from edge devices and mediates when the edge server are not functional. Cloud server collects data from all the edge server middlewares and performs the granulation (Sinanaj et al., 2022), (Haeri et al., 2022). The generated granules with each individual estimated friction cover the entire path of the road, while in comparison each edge server handles a part of the road path. The periodic data collection by cloud server from edge servers helps to identify the operational state of middleware and would coordinate with other edge servers in case of failure of any middleware.

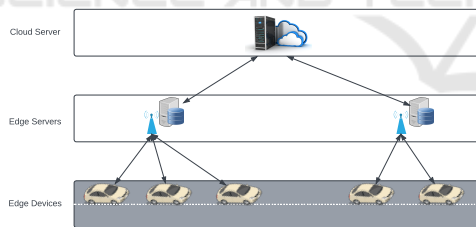


Figure 3: Architecture Components.

With the architectural components discussed (figure 3), the architecture is designed to show the presence of a single edge server for collecting the data from edge devices and also providing responses to edge device requests. When such edge servers are paired together their connectivity is shown for multi-edge server setup.

### 3.4 Single Middleware Setup

An edge server comprises a local database that serves as a repository for data gathered from edge devices. The collected data is processed through granulation algorithms to form granules, which are then stored in

a local database. Additionally, the edge server periodically transmits the data it has collected from the edge devices to a cloud server through a high-bandwidth network connection. Edge devices communicate with the middleware through a wireless IP network. They collect road friction information and generate network packets of a constant length, which contain the location coordinates and friction readings. The IP addresses of all edge servers are pre-configured in the edge devices. For a single middleware system, the edge devices transmit the network packets to the corresponding IP address.

Upon receipt of the information, the edge server performs two actions: (i) it creates granules and provides friction estimates from its local database and (ii) it sends the information collected by the edge device to the cloud, where the granulation process is run on the data collected from the entire road path. At the middleware, granules are generated with hyper-rectangular boundaries and an estimated friction value for all locations within those boundaries. Reduction of the data is seen by using a granule record with boundaries in alternative to a set of actual records covered in it. These granule boundaries, also provide an estimation for location points that are in the vicinity of actual data received from the edge device and happen to be covered by the granule coordinates. Granules are updated in time and maintain relevancy with data received from edge devices.

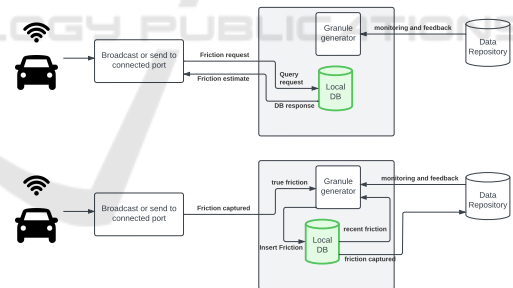


Figure 4: Single middleware setup to generate friction estimates and send friction response.

When the middleware receives a packet request from an edge device, it extracts the location coordinates from the request and queries its database for granules. The estimated friction value of the granule that contains the location point is determined, and all such estimated friction values are grouped together by the middleware to form a response network packet that is sent back to the edge device.(figure 4). Upon receipt of the friction estimates, the edge device uses the information to plan its speed and generates subsequent requests for the remainder of its road journey.



### 3.5 Multi Middleware Setup

When multiple edge servers perform the task of prediction, it is crucial for them to determine the scope of their operations. This enables the edge device to efficiently interact with the relevant edge server to transmit collected data and request information computed by the edge server. For example, in the case of a vehicle on the road that is receiving and transmitting information on friction, the road path is divided into multiple segments based on its length. These segments are then assigned to the edge server middleware, and the vehicle's transmitted information is divided accordingly and passed on to the corresponding edge server.

The road network is divided into segments, each of which is assigned a unique identifier that is given to an edge server. When the middleware receives a packet request that includes the location and segment details, it filters the location point based on the assigned segment identifier. The middleware then collects friction data from vehicles in the designated location points and generates granules to provide friction estimates. Each edge server generates granules for its designated segment locations and also sends the received friction data to the cloud server.

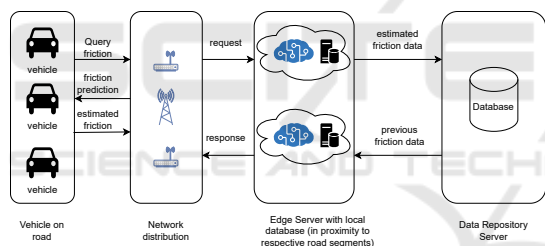


Figure 5: Multiple middleware setup to handle edge device requests.

With all the collected information, the cloud server generates granules (figure 5) for the entire road network and compares these results with those generated by the middleware. The cloud server computation mediates to provide feedback to the edge server when necessary. Feedback achieves similarity of local estimates with cloud friction estimates and relativity of estimates at the shared segment boundary between adjacent edge servers. The distributed computation of edge device data and data reduction using granulation provides less computational and response latency at the edge server. Also, failure for data transmission from the edge server to the cloud server is monitored, to handle failed edge server.

### 3.6 Handover Between Middleware

The edge device is equipped with a list of IP addresses for the middleware network, and as it travels along its route, it broadcasts packets that contain information about detected friction along with a unique segment ID to all edge servers. The edge servers then use the segment ID to filter the packets, with only one edge server collecting the information for a given segment ID and sending a response back to the edge device.

Upon receipt of the response, the edge device uses the IP of the responding middleware to direct its subsequent requests as it moves through the segment. The handover between middlewares as the edge device transitions from one segment to another is facilitated by the implementation of a threshold for the segment boundary. This threshold area signifies the start or end of the segment, and the edge device broadcasts a request as it enters or leaves this area, allowing it to identify nearby middleware servers and establish a connection. The below time flow figure 6 describes the various states of the vehicle when traveling through segments 1 and 2.

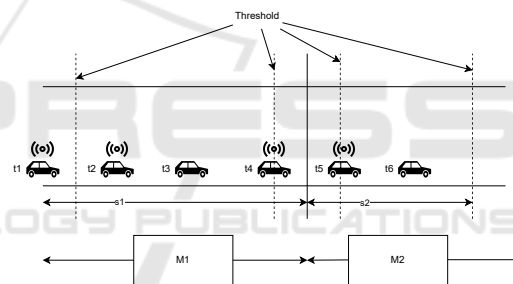


Figure 6: Handover between middleware when the vehicle travels to adjacent segment.

Considering middleware M1 and M2 operating for segments 1 and 2, the vehicle broadcasts its request at time  $t_1$ , indicating its start of the journey. Identifying the threshold, the middleware M1 assigns a threshold flag to vehicle response at time  $t_1$  and also at time  $t_4$  when the vehicle reaches the threshold while exiting segment 1. This allows the vehicle to broadcast the request at time  $t_2$ , which is responded by M1 with flag OFF, indicating vehicle to connect with M1 at  $t_3$  for segment 1 and use M1 IP to request data, while travelling in segment. The vehicle uses the response received from broadcast request at  $t_5$  to connect the middleware M2 at  $t_6$  and use its IP for subsequent requests while traveling in segment 2.

### 3.7 Unresponsive Edge Server

In a multi-edge server system, the consistent updates provided to the cloud by the edge server at equal

intervals facilitate the cloud’s ability to identify any anomalies with the edge server. In the event of an edge server failure, the cloud mediates communication with neighboring edge servers to manage the operations previously handled by the failed node.

In such cases, estimated friction values for the edge device are provided by the adjacent node as the edge device approaches the failed node. Once the edge device travels through the failed edge server segment and reaches the next segment, the detected friction of the failed node is passed on to the next node. The cloud which mediates these middlewares updates the initial edge server with the latest information. The below figure 7, 8 shows a road segment divided into three segments s1, s2, s3 which are handled by M1, M2, M3. When M2 goes unresponsive, the cloud does not receive the time interval updates and detects an anomaly. The cloud server which generated its own granules for segment s2 are passed to edge server M1.

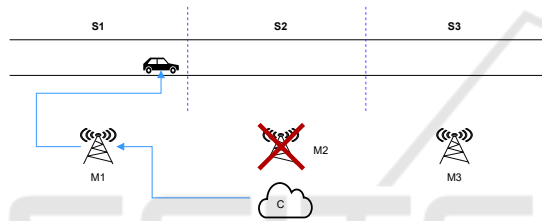


Figure 7: Initial Middleware providing estimated friction for next segment.

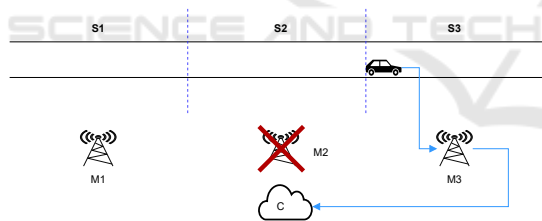


Figure 8: Later Middleware receiving friction experienced from vehicle.

For the edge device entering s2 from s1, M1 detects the threshold flag and sends the estimated friction from s2 granules to the edge device. These friction values are used by edge devices for speed planning while traveling through s2. Once the edge device enters the next adjacent road segment s3, the detected friction for s2 is passed on to edge server M3. M3 sends this information to the cloud, which are converted to as granules of segment s2 and are updated in M1 by the cloud.

## 4 PERFORMANCE

This section shows the experimental results of using edge server architecture over the cloud network architecture. The performance metrics are evaluated for i) computational latency with increasing road length and ii) effect of increasing edge device requests.

### 4.1 Experimental Setup

The proposed experiments are developed with python 3.11 and performed on a windows 10 Enterprise machine with 32 GB RAM and clock speed of 3.20GHz. Dataset: A simulated road friction data is generated that shows variation in friction values in different road locations and contains attributes of East, North, time, friction estimate, and friction true (Gao et al., 2022). The location of the vehicle is provided by East and north attributes and the time attribute in seconds provides the simulation time from zero seconds. The friction true is actual friction on road, while estimate friction is vehicle-measured friction which is actual friction added with the white noise of magnitude of 0.3.

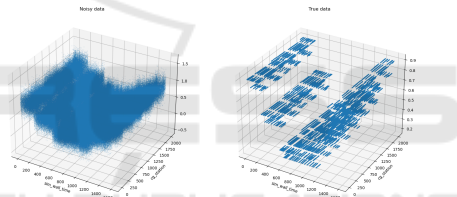


Figure 9: Friction Variation - a) with White Noise b) True Friction.

### 4.2 Computational Latency for Increasing Road Length

#### 4.2.1 Dual Edge Server Setup

Considering a road path, whose friction information is operated by two edge servers, we divide the road path into two segments, allocating equal area paths for both edge servers. With friction information received from vehicles traveling along the road path, both edge servers generate granules in their defined road length.

We simulate the constant rate of vehicle requests that are traveling on different locations of the road and calculate the time taken by the edge server to respond with friction estimate. Time taken is averaged among multiple runs for better estimate. Further, we replace edge servers with a single data repository server, which handle the granulation and friction response for entire road path and compute response time of data repository server to respond to same amount of requests handled by each edge server.

From the plot figure 10, when the data repository server and dual edge servers handle 24000 queries, we see with the increase in road length, the average data repository response time is larger than the edge server. With the increase of road length, the response time by data repository extended by 1.864% per unit road length, while the dual edge server response time increased by 0.943%

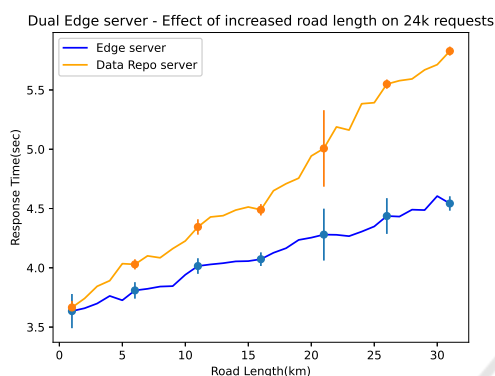


Figure 10: Dual edge server Query response with variance scale.

### 4.2.2 Dynamic Edge Server Allocation

For the dual edge server, with the increasing road length the range of the edge server is extended. For our next experiment, we set the edge servers to operate on a fixed length of the road. For the equal incremental road length range, we calculate the response time of a data repository server handling entire road path and edge server handling constant road path.



Figure 11: Dynamic Middleware Query response with variance scale.

From the plot figure 11, we see that average response time increases with road length when compared with the edge server. When such edge servers are placed along the road network, the overall response time is greatly reduced improving speed planning. With the implementation of a dynamic middle-

ware system, the increase in response time for the data repository server is 1.84%, and per unit road length increase in response time for the dynamic edge server is 0.25%

### 4.3 Effect of Increasing Edge Device Requests

When more number of vehicles travel on the road network, the increased vehicle requests adds more delay in query response by the computing server. To check the effect of increasing incoming requests on the server, we consider a multi-edge setup, where the entire road path is divided among 30 edge servers.

We simulate the increasing range of vehicle requests by equal amounts in the edge server and cloud server. The response time is considered from fetching relevant granules that cover the location point and computing the average from friction attribute of the fetched granules. These response times are added up for all the queried location points provided as response time for a query set. Such response times are evaluated multiple times to provide a better average friction estimate.

From the plot figure 12, the increase in response time is drastic in the cloud server when compared to the edge server. For increased vehicle requests to 40000, the multi-edge server achieved a 42.358% reduction in response time, in comparison with the cloud server.

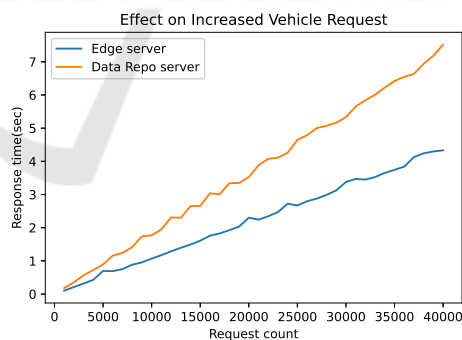


Figure 12: Effect of Increased Vehicle requests.

## 5 CONCLUSION

In this paper, we present a distributed edge-cloud model that collaborates with mobile edge devices, such as vehicles on the road.

Utilizing the Allan variance averaging technique, which determines the optimal split for reducing data to be processed, we maintain accurate friction estimation results while processing on the edge server. Our

proposed architecture demonstrates that a coordinated edge server system delivers continuous responses as vehicles travel along the road. The edge server's failure handling mechanism effectively addresses real-time communication breakdowns. We illustrate that the coordinated edge server system enables faster data processing, lower latency, and higher throughput for requests compared to the cloud-mediated transmission model.

In future work, we aim to expand our research on the feedback system between cloud and edge servers. We plan to adapt the cloud to identify variations in friction estimates generated over time and use this information to determine the feedback interval with the edge servers.

## ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under grant no. CNS-1932509 "CPS: Medium: Collaborative Research: Automated Discovery of Data Validity for Safety-Critical Feedback Control in a Population of Connected Vehicles"

## REFERENCES

- Allan, D. W. (1966). Statistics of atomic frequency standards. *Proceedings of the IEEE*, 54(2):221–230.
- Barrachina, J., Garrido, P., Fogue, M., Martinez, F. J., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2013). Road side unit deployment: A density-based approach. *IEEE Intelligent Transportation Systems Magazine*, 5(3):30–39.
- Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The  $r^*$ -tree: An efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331.
- Cai, H. and Lin, Y. (2008). A roadside its data bus prototype for intelligent highways. *IEEE Transactions on Intelligent Transportation Systems*, 9(2):344–348.
- Gao, L., Beal, C., Mitrovich, J., and Brennan, S. (2022). Vehicle model predictive trajectory tracking control with curvature and friction preview. *IFAC-PapersOnLine*, 55(24):221–226. 10th IFAC Symposium on Advances in Automotive Control AAC 2022.
- Haeri, H., Soleimani, B., and Jerath, K. (2022). Optimal moving average estimation of noisy random walks using allan variance-informed window length. In *2022 American Control Conference (ACC)*, pages 1646–1651.
- ISO/IEC (2020). Tr 30164:2020 - Internet of Things (IoT) - Edge Computing. Technical report, ISO/IEC.
- Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95.
- Kurunji, S., Ge, T., Fu, X., Liu, B., and Chen, C. X. (2013). Optimizing communication for multi-join query processing in cloud data warehouses. *Int. J. Grid High Perform. Comput.*, 5(4):113–130.
- Li, J., Luo, G., Cheng, N., Yuan, Q., Wu, Z., Gao, S., and Liu, Z. (2018). An end-to-end load balancer based on deep learning for vehicular network traffic control. *IEEE Internet of Things Journal*, 6(1):953–966.
- Mendki, P. (2018). Docker container based analytics at iot edge video analytics usecase. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4.
- Ou, C.-H., Wu, B.-Y., and Cai, L. (2019). Gps-free vehicular localization system using roadside units with directional antennas. *Journal of Communications and Networks*, 21(1):12–24.
- Panahandeh, G., Ek, E., and Mohammadiha, N. (2017). Road friction estimation for connected vehicles using supervised machine learning. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1262–1267.
- Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.
- Sinanaj, L., Haeri, H., Maddipatla, S. P., Gao, L., Pakala, R., Kathiriya, N., Beal, C., Brennan, S., Chen, C., and Jerath, K. (2022). Granulation of large temporal databases: An allan variance approach. *SN Computer Science*, 4(1):7.
- Tang, S., Chen, B., Iwen, H., Hirsch, J., Fu, S., Yang, Q., Palacharla, P., Wang, N., Wang, X., and Shi, W. (2021). Vecframe: A vehicular edge computing framework for connected autonomous vehicles. In *2021 IEEE International Conference on Edge Computing (EDGE)*, pages 68–77.
- Van Rijmenam, M. (2017). Self-driving cars will create 2 petabytes of data, what are the big data opportunities for the car industry.
- Velmurugadass, P., Dhanasekaran, S., and Sasikala, S. (2021). The cloud based edge computing with iot infrastructure and security. In *2021 International Conference on Computational Performance Evaluation (ComPE)*, pages 030–034.
- Wang, Y., Sheng, M., Wang, X., Wang, L., and Li, J. (2016). Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications*, 64(10):4268–4282.