FogROS2-PLR: Probabilistic Latency-Reliability For Cloud Robotics

†Kaiyuan Chen[1,2], Nan Tian[2], Christian Juette[2], Tianshuang Qiu[1,2],
Liu Ren[2], John Kubiatowicz[1], and Ken Goldberg[1,3]

***Abstract*— Cloud robotics enables robots to offload computationally intensive tasks to cloud servers for performance, cost, and ease of management. However, public networks and cloud computing infrastructure are not designed for reliable timing guarantees, and are prune to fluctuating Quality-of-Service (QoS). In this work, we formulate an impossibility triangle theorem for: Latency reliability, Singleton server, and Commodity hardware. The LSC theorem suggests that providing replicated servers with uncorrelated failures can exponentially reduce the probability of missing a deadline. We present FogROS2-Probabilistic Latency Reliability (PLR) that uses multiple independent network interfaces to send requests to replicated cloud servers and uses the first response back. We design routing mechanisms to discover, connect, and route through non-default network interfaces on robots. FogROS2-PLR optimizes the selection of interfaces to servers to minimize the probability of missing a deadline. We conduct a cloud-connected driving experiment with two 5G service providers, demonstrating FogROS2-PLR effectively provides smooth service quality even when one of the service providers experiences low coverage and base station handover. We use 99 Percentile (P99) latency to evaluate anomalous long-tail latency behavior. In one experiment, FogROS2-PLR improves P99 latency by up to 3.7x compared to using one service provider. We deploy FogROS2-PLR on a physical Stretch 3 mobile robot performing an indoor human-tracking task. Even in a fully covered Wi-Fi and 5G environment, FogROS2-PLR improves the responsiveness of the robot reducing mean latency by 36% and P99 latency by 33%. Code and supplementary can be found on website `https://github.com/data-capsule/rt-fogros2`.**

I. Introduction

The complexity of robotic algorithms [1, 2] and models [3, 4] often surpasses the computing capabilities of onboard hardware of robots. State-of-the-art Large Language Models (LLM) [5, 6], Vision Language Models (VLM) [7–11] and Vision-Language-Action (VLA) [12–17] are large and almost always hosted in cloud environment. FogROS2 [18] is an open-source Cloud Robotics framework that deploys unmodified compute-intensive algorithms in Robot Operating System 2 (ROS2). However, Cloud robotics infrastructures are typically implemented to prioritize cost and resource efficiency, often sharing network or compute infrastructure among multiple robots and heterogeneous workloads. However, as network quality of service (QoS) between robots and the cloud varies, we show providing *Latency-Reliable* cloud robotics services on *Commodity*[1] infrastructure is impossible with a single cloud server setup. We introduce FogROS2-PLR that simultaneously uses independent robot network interfaces and independent cloud servers (Figure 1). FogROS2-PLR enhance performance against variable network QoS and infrastructure unavailability and improve the responsiveness of cloud robotics despite the following limitations:

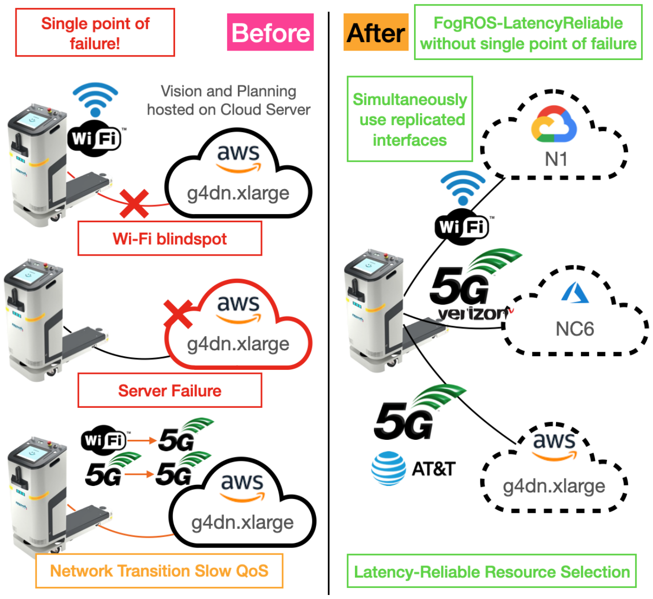

Single point of failure!
Before
After
FogROS-LatencyReliable without single point of failure
Vision and Planning hosted on Cloud Server
Wi Fi
aws
g4dn.xlarge
Wi-Fi blindspot
Server Failure
5G
Network Transition Slow QoS
Simultaneously use replicated interfaces
N1
verizon
NC6
AT&T
Latency-Reliable Resource Selection

Fig. 1: **FogROS2-PLR Use Case.** A mobile robot in a warehouse connects to the cloud for vision, planning, and coordination. A smooth connection is required for safety and responsiveness. **(Left)** Conventional cloud robotics is subject to a single point of failure. In the top and middle, network or server failure leads to a complete breakdown of the system. At the bottom, transition to an alternative network or server at slowdown leads to QoS degradation. **(Right)** Instead, FogROS2-PLR provides a fault-tolerant solution that deploys unmodified ROS2 applications to multiple low-cost cloud servers, making cloud-robotics applications resilient to individual server termination and network slowdowns.

(A) *Latency-Unreliable Computational Infrastructure*: The cloud allows for flexible use of computational resources. For instance, systems can be oversubscribed by allocating fewer resources than the total required by all robots, assuming that not all robots demand resources simultaneously. This approach enhances resource utilization, but if too many robots access resources simultaneously, it can lead to latency degradation or even failures.

(B) *Latency-Unreliable Network Infrastructure*: The net-

[1]Department of Electrical Engineering and Computer Science
[2]Robert Bosch Research and Technology Center North America, Sunnyvale, CA, USA
[3]Department of Industrial Engineering and Operations Research
[1,3]University of California, Berkeley, CA, USA
†For correspondence and questions: kych@berkeley.edu

[1] We borrow the term ‘commodity’ from The Google File System [19]. The word is widely used in network research [20–22] to describe cheap, latency unreliable and error-prone compute, network and storage infrastructure

work infrastructure connecting robots to the cloud can experience varying levels of reliability. In cloud robotics, the network's quality of service (QoS) is crucial for maintaining consistent communication and performance. However, fluctuations in network bandwidth or latency can occur due to factors such as network congestion, physical distance, and network outages.

(C) *System Dynamism*: Latency fluctuations can sometimes arise not from the infrastructure itself but from the underlying mechanisms. For instance, in a 5G network hosted by a base station, handovers are required to maintain connectivity as a vehicle moves from one cell to another. In addition, a mobile robot in a warehouse may need to switch between an indoor Wi-Fi connection and a 5G cellular connection. The operating system must detect signal reduction or link breakdown and then initiate a switch to an alternative network. This process can take anywhere from 100 milliseconds to 10 seconds [23]. In both scenarios, frequent network switching can cause service interruptions, while delayed switching can result in slow quality of service.

Specially designed real-time operating systems [24–26] and network communication protocols [27–30] rely on dedicated resources to mitigate latency unreliability. However, deploying such systems is expensive, and requires overprovisioning network and compute resources. In addition, in public networks, achieving dedicated resource allocation might not even be possible due to policy decisions, such as those surrounding net neutrality. Public networks are designed to treat all data equally, which limits the ability to prioritize certain types of traffic or allocate specific resources to a single application. This restriction makes it challenging to implement real-time systems that rely on guaranteed low-latency performance in such environments.

Alternatively, we formulate probabilistic latency-reliable cloud robotics operating on unreliable commodity infrastructure. We find that providing replicated resources with uncorrelated failures can reduce the failure probability exponentially. We propose FogROS2-PLR, a cloud robotics framework that uses multiple independent networks for the robot and failure-independent compute resources to achieve probabilistic latency reliability on commodity cloud infrastructure. FogROS2-PLR discovers, connects, and simultaneously routes messages through multiple non-default network interfaces on robots to identical services deployed in multiple cloud data centers and uses the first response received from the replicated services. This model significantly increases the probability of getting timely responses as long as at least one replica and the network remain operational and responsive. FogROS2-PLR optimizes the selection of interfaces to cloud-deployed robotics algorithms by minimizing the probability of deadline misses.

We evaluate FogROS2-PLR with a cloud-connected driving experiment with two 5G service providers, demonstrating FogROS2-PLR effectively provides smooth quality of service even if one of the service providers experiences low coverage and base station handover. In the experiment, FogROS2-PLR improves anomalous long tail P99 (99 Percentile) latency by up to 3.7x. We deploy FogROS2-PLR on a physical Stretch 3 robot with an indoor human-tracking task. Even in a fully covered Wi-Fi and 5G environment, FogROS2-PLR the responsiveness of the robot by reducing 36% of mean latency and 33% of anomalous long-tail latency.

This paper makes five contributions: (1) Formulation of Probabilistic Latency-Reliability for cloud robotics; (2) LSC Theorem that latency reliability, singleton deployment, and commodity infrastructure cannot co-exist; (3) FogROS2-PLR, a cloud robotics framework that uses independent network interfaces and cloud services for latency reliability; (4) Algorithm to determine the optimal choice of network interface-server combination to maximize latency reliability; (5) Evaluation of FogROS2-PLR on human tracking and autonomous driving applications.

II. Related Work

Cloud and Fog Robotics: The use of cloud computing resources for robots conceptualized as cloud robotics [31], has become increasingly relevant as large models. These computationally demanding models/ algorithms have a wide range of applications in robotics, such as visual perception [3, 4]). Following the Fog Computing paradigm [32], Fog Robotics [33] utilizes edge resources to improve performance, enabling the viability of cloud computing for a multitude of robotics applications [1, 2, 34, 35]. FogROS2 [36] is a cloud robotics framework officially supported by ROS2 [37]. The extension of FogROS2, FogROS2-SGC [38] enables secure communication between distributed ROS2 robot nodes. of this work has addressed the questions of connectivity, latency, and cost.

Latency Sensitive Cloud and Fog Infrastructure: Edge or Cloud real-time systems [24, 25] on commodity hardware typically assume the networking has reliable latency. For example, Edge-RT[26] makes the best effort to guarantee the deadline after the packet is received. However, providing low-latency networking in wireless network settings for automation and robotics requires special hardware such as radio frequency, wireless channels, and environment assumptions [27–30]. FogROS2-PLR recognizes that deploying such systems can be expensive and aims for a practical and general latency reliable cloud robotics framework.

Latency Sensitive Cloud Robotics via Redundancy: Existing work uses *redundancy* by duplicated machines to prevent failures of one machine. Schafhalter, Kalra, Xu, Gonzalez, and Stoica [39] improves the responsiveness of autonomous vehicles by performing operations on both vehicle and cloud. FogROS2-LS (Latency Sensitive) [40] uses replicated communication and compute resources so that a robot can flexibly connect to using one of many servers, but the system takes time to discover and recover from faults by switching to another server that meets latency requirements. Wu, Zheng, Chen, Wang, and Cao [41] statistically models and detects anomalous time series events. Prior work also designed applications to fault tolerance environments with spot VMs, such as deep learning [42].

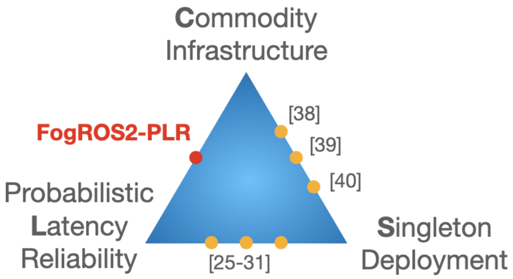


Fig. 2: **Impossibility Triangle of LSC Theorem** Among probabilistic Latency Reliability, Singleton deployment, and Commodity infrastructure, a cloud robotics system can have at most two of these three properties. We characterize FogROS2-PLR and its related work on the edges of the impossibility triangle.

III. Impossibility Triangle of Probabilistic Latency-Reliable Cloud Robotics

We consider a closed-loop, deterministic and stateless cloud robotics task. The robot r connects independently with the independent server(s) S that hosts the cloud robotics service. All together they form a network graph $G = (V,\ E)$, where V is the set of nodes (including r, servers S , and E is the set of edges representing the communication links. One can deploy multiple servers, so we use $\mathscr{P}_{rS}$ to denote all the paths that the robot connects to possible servers, and a robot r can connect to a cloud server $s \in S$ with a bidirectional path $p_{rs} \in \mathscr{P}_{rS}$.

The robot sends a request and awaits the server s's response through path $p_{r \to s}$ with the round-trip latency $L_{p_{rs}}$, which is the sum of the network latency from robot r to server s on the path $p_{r \to s}$ and the processing latency of the server L_s.

Given a latency deadline L_D, the cloud robotics service is defined as **latency reliable** if the probability of latency L exceeding a threshold $L_D + \delta$ is less than ε for all $\varepsilon > 0$ and $\delta > 0$. Formally,

$$P(L \leq L_D + \delta) < \varepsilon$$

Typically, Cloud Robotics is a **singleton deployment** such that there is only one server s that hosts the service ($S = \{s\}$) and to server s, exactly one path, p_{rs}, is selected to connect r to s at any given point in time. In this case, $L = L_{p_{rs}}$.

The robot and cloud are deployed with latency-unreliable **commodity infrastructure** such that each path $p_{rs} \in \mathscr{P}_{rs}$ from robot r to server s follows a latency probability distribution $P(L_{p_{rs}})$ that for some $\delta_0 > 0,\ \varepsilon_0 > 0$,

$$P(L_{p_{rs}} > L_D + \delta_0) \geq \varepsilon_0, \tag{1}$$

With the definitions, we formulate the following theorem:

LSC Theorem Among Latency Reliability, Singleton deployment, and Commodity infrastructure, a cloud robotics system can have at most two of these three properties.

Proof Sketch We show that assuming any two of three properties implies that the third property cannot be achieved.

Case 1: Singleton Deployment and Commodity Infrastructure With singleton deployment, one path, p_{rs} is selected with latency $L = L_{p_{rs}}$. If the infrastructure is latency unreliable, the latency L on this single path does not meet the reliability requirement in Equation 1 with ε_0 and δ_0. Thus, latency reliability cannot be guaranteed if the infrastructure is unreliable and there is only one path. This directly contradicts the latency reliability property.

Case 2: Singleton Deployment and Latency Reliability With singleton deployment, there is exactly one path for communication, and achieving latency reliability implies this path must consistently meet the latency requirements. However, if the infrastructure uses commodity hardware it is by definition latency unreliable. Thus the single path fails to meet the latency requirements, making latency reliability impossible. This creates a contradiction and thus cannot use generic infrastructure.

Case 3: Latency Reliability and Commodity Infrastructure Commodity infrastructure always have a non-negligible probability in the latency requirement in Equation 1 with ε_0 and δ_0. Therefore, the only way to achieve latency reliability in such an infrastructure is to use more than one independent network and server.

Theoretical Implication To achieve latency reliability, one can choose the (singleton deployment, reliability) edge in the triangle (Fig. 2) to use dedicated specialized infrastructure with real-time guarantees. However, deploying such infrastructure can be both resource inefficient and expensive [24–27].

Alternatively, one can choose (commodity hardware, reliability) edge by providing independent server redundancy $|S| > 1$ and independent network path redundancy that $|\mathscr{P}_{rS}| > 1$ on top of an unreliable infrastructure. Suppose the system consists of n possible servers $\{s_i\}_{i=1}^{n}$ and to server s_i, m_i independent network paths $\{p_{ij}\}_{j=1}^{m_i} \subseteq \mathscr{P}_{rs}$. For a single path p_{ij} to server S_j, the probability that its latency $L_{p_{ij}}$ exceeds the threshold $L_D + \delta$ is:

$$P(L_{p_{ij}} > L_D + \delta) = \varepsilon_{ij}.$$

The independence of paths and servers implies that the total system latency L is determined by the best combination of path and server. The probability that all combinations exceed $L_D + \delta$ is:

$$P(L > L_D + \delta) = \bigcap_{i=1}^{m} \bigcap_{j=1}^{n} P\left(L_{p_{ij}} > L_D + \delta\right)$$

The product decreases exponentially with the number of paths m and servers n $(\varepsilon')^{mn} < \varepsilon$ for $\varepsilon > 0$. Specifically,

$$\lim_{m \to \infty, n \to \infty} \bigcap_{i=1}^{m} \bigcap_{j=1}^{n} \varepsilon_{ij} = 0$$

This shows that increasing the number of independent paths and servers significantly reduces the probability of exceeding the latency threshold.

Practical Implication The assumption of independence is important because it allows us to multiply the probabilities of individual paths to determine the overall probability of meeting latency requirements. However, in practice, achieving the independence is challenging because users typically cannot control the routing of packets once they enter the

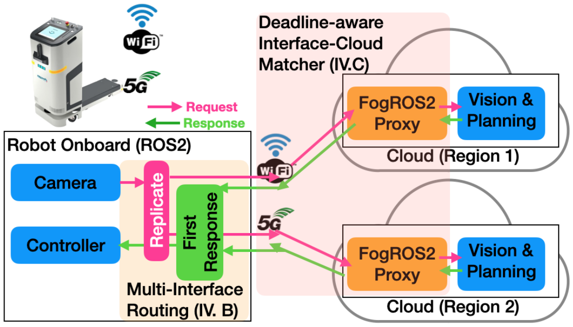


Fig. 3: **System Overview of FogROS2-PLR** FogROS2-PLR transparently proxies ROS2 communication. It sends requests through multiple network interfaces (such as 5G and Wi-Fi) to replicated Cloud VMs. It uses the first response back to the robot.

Internet service providers infrastructure. To approximate independence, we can use multiple network interfaces on the robot, and connect them to different and independent Internet Service Provider (ISP) backbone network. For example, a robot connecting to a Wi-Fi network provided by Comcast is unlikely to experience correlated failures with its connection to a 5G cellular network with AT&T.

In summary, we make the following assumptions for practical deployment: 1) Different network backbones do not experience correlated failures. 2) Onboard processing overhead of an additional network interface is negligible. 3) The robots stay within the planned probabilistic reliability distribution.

IV. FogROS2-PLR System Design

Figures 3 show an overview of how FogROS2-PLR achieves latency reliability. All the designs of FogROS2-PLR are implemented within a proxy that sits between ROS2 and our custom network protocol. This proxy allows unmodified ROS2 applications to seamlessly integrate with our framework. FogROS2-PLR replicates requests over multiple independent network interfaces to cloud servers at different regions and routes the first response back to the robot. Different interface selections and independent regions are used to reduce the correlated failures. Section IV.A presents how FogROS2-PLR matches the interface to the cloud server to minimize the probability of deadline miss. Section IV.B discusses how FogROS2-PLR discovers and connects with the cloud with multiple network interfaces.

A. Deadline-Aware Interface-Cloud Matcher

To determine *which* interface should be used to connect to *which* cloud server, FogROS2-PLR minimizes the probability of missing the latency deadline by the deadline-aware interface-cloud matcher.

Suppose the system consists of n possible servers $\{s_j\}_{j=1}^n$ and on the robot, there are m interfaces that are able to connect to those servers. We use a binary decision variable x_{ij} denoting if we should use interface i to connect with server s_j,

$$x_{ij} = \begin{cases} 1 & \text{if interface } i \text{ is connected to server } s_j \\ 0 & \text{otherwise} \end{cases}$$

and the latency ε_{ij} is probability of missing the deadline using interface i to connect with server s_j.

We write the objective function as:

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} \varepsilon_{ij} \cdot x_{ij}$$

subject to constraints:

1) Binary decision variables: $x_{ij} \in \{0,1\} \quad \forall i \in \{1,2,\ldots,m\}, \forall j \in \{1,2,\ldots,n\}$
2) Each robot interface should connect to exactly one server: $\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in \{1,2,\ldots,m\}$
3) The number of connections to each server must not exceed its capacity: $\sum_{i=1}^{m} x_{ij} \leq 1 \quad \forall j \in \{1,2,\ldots,n\}$

This problem can be solved using integer programming [43] to find the optimal assignment of robot interfaces to servers that minimize latency while satisfying the constraints. L_{ij} can be measured by estimating separately with the computational time the compute and network time. This reduces the overhead of profiling. For example, if the system only uses one class of cloud servers for computation, only the network data needs to be collected by permuting the network interfaces to the cloud servers.

B. Multi-Interface Connectivity

Figure 4 shows the workflow of FogROS2-PLR on handling requests with fault tolerance. At the connection setup phase, FogROS2-PLR creates a thread for every available interface available to the robot and creates a socket that forces it to bind to the interfaces even if the interfaces are not default. Getting the cloud-connectable network addresses from non-default interfaces is challenging, because some local or virtual interfaces, such as Virtual Private Network, are private to the robot. We use a cloud-deployed Session Traversal Utilities for NAT (STUN) protocol [44] to get the cloud-accessible network IP address. The thread is canceled if it is unable to connect to address STUN service for a period of time. The thread advertises itself to the discovery service; the discovery service runs a custom protocol to exchange the addresses of discovered addresses similar to FogROS2-SGC [38]. Both address service and discovery service can be held in public Internet or in private network settings. These services are solely used for establishing connectivity, and no application data passes through the server. Therefore, a server failure does not result in a system failure.

When the robot sends a ROS2 request, FogROS2-PLR's proxy on the robot intercepts it, and extracts its serialized request and a unique identifier from the ROS2 middleware layer (rmw). The proxy then stores this identifier with a handle and replicates the request to the cloud through multiple interfaces. Deadline-Aware Interface-Cloud Matcher constantly updates the mapping of Interface-Cloud. The cloud proxy executes the corresponding cloud-based ROS2 service,

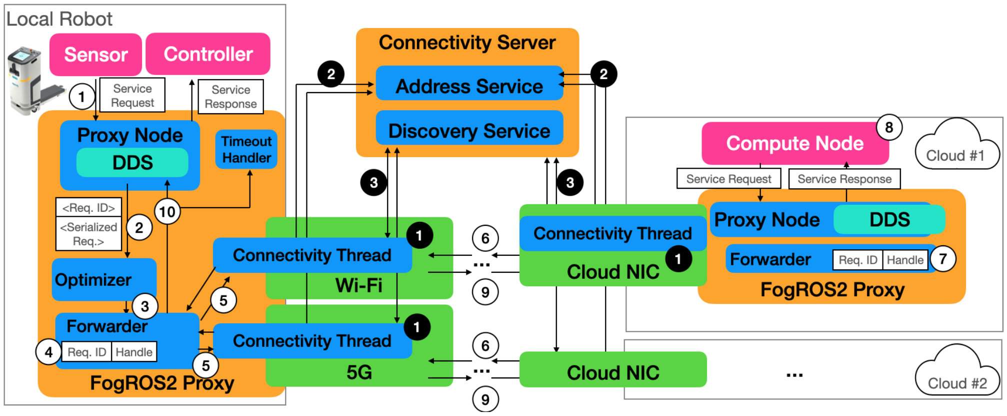


Fig. 4: **Workflow Diagram of FogROS2-PLR** On Setup (Black circle), **(1)** FogROS2-PLR proxy instantiates threads per interface (5G, Wi-Fi, and Cloud Network Interface Card (NIC) in Green) per service to handle communication ; **(2)** The thread communicates with a centralized connectivity server. The connectivity server runs STUN protocol to get the public IP address of the given interface; **(3)** The thread advertises the service-address binding to a centralized discovery service. The discovery service facilitates the cloud and robot to discover each other. To handle an incoming request from the robot (White circle), **(1)** The ROS2 application sends a request to FogROS2-PLR proxy; **(2)** The proxy retrieves a unique request ID from ROS2 Data Distribution Service (DDS) and raw request in bytes; **(3)** The request goes through optimizer that determines network interface and cloud server mapping; **(4)** The forwarder registers the unique request ID from rmw with a callback for response and a callback for timeout; **(5)** The forwarder replicates and sends the request to the corresponding server specified by the optimizer mapping; **(6)** The message is routed in the failure-independent networks from the robot to the cloud; **(7)** The FogROS2-PLR proxy converts the request to a regular ROS2 service request; **(8)** The cloud proxy invokes the cloud ROS2 service and gets the response; **(9)** The response is forwarded back the robot; **(10)** The robot proxy uses the first received response from replicated network interfaces and servers, and returns with a regular ROS2 service response to the application. It can optionally invoke a timeout callback if the response is not returned promptly.

computes the result, and sends the response back to the robot's proxy. The robot's proxy then verifies the response using the unique identifier. If the identifier matches, the request is marked as completed and the response is sent to the robot; otherwise, it is discarded as a duplicate.

Multi-Interface Asynchronous Proxy While each interface independently sends packets, FogROS2-PLR must ensure efficient internal processing. To achieve this, FogROS2-PLR creates a separate thread for each interface to handle packet processing and potential retransmission in case of packet loss. Maintaining efficiency is crucial; inefficient packet processing compromises the independence property of additional network interfaces: processing on one interface could block others, and the sequence of packet sending would violate the independence assumption. FogROS2-PLR uses asynchronous operations throughout the packet processing pipeline, allowing the proxy to continue processing packets from other interfaces without delay. Additionally, in FogROS2-PLR, packets are processed with a zero-copy property, meaning that the packet data is not duplicated in memory during processing. This minimizes the overhead associated with using multiple interfaces, preserving efficiency and maintaining the integrity of the independence assumption.

V. Experiments

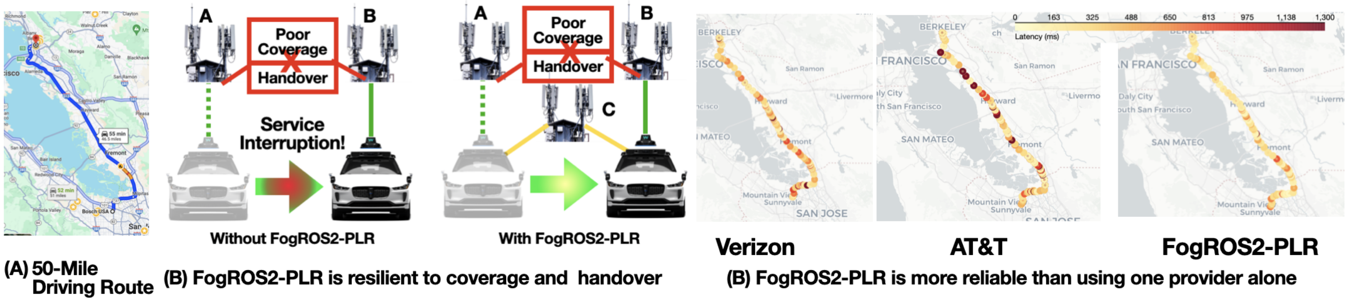
We evaluate the latency reliability of FogROS2-PLR with (1) a simulated robot vision with semantic segmentation with two Wi-Fi interfaces, (2) a cloud operation in a high mobility driving setting with two 5G interfaces, (3) an indoor human tracking with Wi-Fi and 5G. We quantify *long-tail* anomalous latency faults with 99 Percentile (P99) latency, the runs with the top 1 % latency.

A. Case Study: Cloud Operation with High Mobility

Motivation Autonomous vehicles, such as Waymo, Cruise, and the upcoming Robotaxi use cloud operation to remotely take control in emergencies, such as when a car gets stuck and the onboard algorithms cannot resolve the issue. This cloud operation may involve a teleoperator, a remote data server or a more sophisticated model [45]. Our study focuses on how to maintain high-quality service in scenarios involving high mobility.

Setup We conducted a 50-mile drive along a standard U.S. highway from Sunnyvale, CA to Berkeley, CA, over a 90-minute period. The car's speed was maintained by Tesla's Full Self Driving. During the drive, we streamed 1280x820 resolution images as JPEG from an Intel RealSense 435i camera connected to a laptop. The laptop was connected via physical cables to two different 5G hotspots provided by Verizon and AT&T. We emulated the decision-making process of a cloud operator using object detection with YOLOv8 [46]. The end-to-end latency was measured from the moment an image was captured to the time a response was received.

Results Figure 5 compares FogROS2-PLR with the use of a single service provider. In comparison, AT&T had higher mean and P99 latency in some extreme cases. FogROS2-PLR effectively leveraged both service providers, offering more reliable latency. FogROS2-PLR improves P99 latency of AT&T and Verizon respectively by 3.7x (4829.35ms vs 1303.00ms) and 2.4x (3208.04 vs 1303.00). It improves mean latency (289.44ms) than AT&T (791.02ms) and Verizon (550.28ms). We observed a slight failure case near Fremont, CA, where both service providers exhibited higher latency for some requests, likely due to coverage issues relative to the car's speed.



(A) 50-Mile Driving Route (B) FogROS2-PLR is resilient to coverage and handover (B) FogROS2-PLR is more reliable than using one provider alone

Fig. 5: **Case Study (A) High Mobility Cloud Operation with FogROS2-PLR** (A) 50-mile Driving route from Sunnyvale, CA to Berkeley, CA with 50 miles. (B) In the mobility test, the car experiences a coverage blind spot of 5G service providers. 5G *handover*: the base station passes the connectivity to the next base station at mobility. Both coverage and handover lead to QoS degradation. FogROS2-PLR prevents such interruption by an independent 5G service provider. (3) While relying on a single service provider leads to QoS fluctuations, FogROS2-PLR demonstrates smooth connectivity by using multiple 5G networks. FogROS2-PLR improves P99 latency of AT&T and Verizon respectively by 3.7x and 2.4x; it improves mean latency by respectively 2.7x and 1.9x.

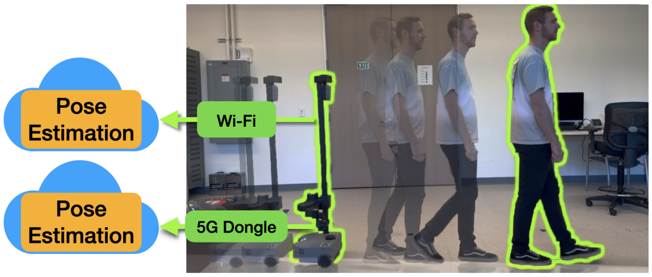


Fig. 6: **Case Study (B) Human Following Robot with FogROS2-PLR in a fully-covered 5G and Wi-Fi environment Setup** We deployed FogROS2-PLR on the Hello Robot Stretch 3 Mobile Manipulator. Stretch Robot connects to two cloud servers running human pose estimation via both 5G and Wi-Fi with FogROS2-PLR.

B. Case Study: Human Following Mobile Robot

Setup: We deploy FogROS2-PLR on the Hello Robot Stretch 3 Mobile Manipulator, running off-the-shelf Ubuntu 22.04 and ROS2 Humble. The mobile manipulator was tasked with following a human along an indoor route around an office. We programmed the tracking and motion based on [46]. The robot was connected to both a Wi-Fi dongle with a 5G hotspot and the office's Wi-Fi network.

Results Figure 7 shows the result of FogROS2-PLR, which improves the P99 long tail latency by 33% (160.00ms vs 193.52ms) even in an environment with full coverage of Wi-Fi and 5G. FogROS2-PLR can consistently improve the average latency by 36% (71.39ms vs 97.19ms).

Baseline Failure Case Throughout the route, both the 5G and Wi-Fi networks maintained full coverage, despite the varying signaling strength. We attempted to set up a scenario where one network's coverage would drop, requiring the Operating System to switch between Wi-Fi and 5G. Unfortunately, ROS2, using either CycloneDDS (the default DDS before ROS2 Humble) or FastDDS (the default DDS for ROS2 Humble), is unable to switch from the original default interface. This is a known limitation [47].

VI. Conclusion

In this work, we formulate probabilistic latency reliability on latency-unreliable cloud robotics servers and use failure-independent networks and cloud resources to achieve PLR. The evaluation shows FogROS2-PLR can reduce P99 latency by up to 3.7 times in a realistic driving experiment.

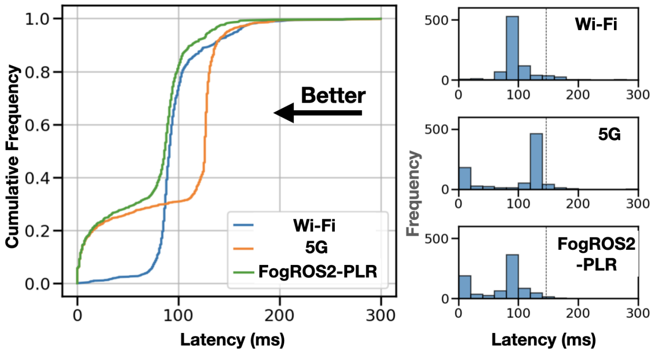


Fig. 7: **Case Study (B) Human Tracking with FogROS2-PLR Results** (3) Results of FogROS2-PLR. (Left) The comparison of Wi-Fi and 5G latency distributions shows that Wi-Fi generally offers better mean latency, while 5G excels in low-latency scenarios but exhibits less reliable overall latency. FogROS2-PLR effectively combines the strengths of both interfaces, optimizing latency performance. FogROS2-PLR is able to take advantage of both of the interfaces. (Right) The Figure shows the histogram of latency with a dashed line marking P99 of FogROS2-PLR compared to other options demonstrating that FogROS2-PLR consistently outperforms using either network alone.

In future work, we plan to enhance the real-time capabilities of FogROS2-PLR by deploying in a local or edge real-time environment and incorporating a fallback mechanism for handling missed deadlines. Additionally, we will investigate strategies to reduce the cost of using replicated network resources while maintaining high reliability and availability. For example, our earlier work, FogROS2-FT, shows Cloud Spot Virtual Machines can effectively reduce operating costs. In future work, we will explore adaptive scheduling algorithms that can reduce the cost while maintaining PLR.

References

[1] J. Ichnowski *et al.*, "Fog robotics algorithms for distributed motion planning using lambda serverless computing," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 4232–4238.

[2] A. K. Tanwani, N. Mor, J. Kubiatowicz, J. E. Gonzalez, and K. Goldberg, “A fog robotics approach to deep robot learning: Application to object recognition and grasp planning in surface decluttering,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2019, pp. 4559–4566.

[3] A. Kirillov *et al.*, “Segment anything,” *arXiv preprint arXiv:2304.02643*, 2023.

[4] A. Rashid *et al.*, “Lifelong lerf: Local 3d semantic inventory monitoring using fogros2,” *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024.

[5] H. Touvron *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

[6] J. Achiam *et al.*, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.

[7] “Gpt-4v(ision) system card,” 2023.

[8] Google, “Gemini: A family of highly capable multimodal models,” *arXiv preprint arXiv:2312.11805*, 2023.

[9] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” *Advances in neural information processing systems*, vol. 36, 2024.

[10] J.-B. Alayrac *et al.*, “Flamingo: A visual language model for few-shot learning,” *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.

[11] D. Driess *et al.*, “Palm-e: An embodied multimodal language model,” in *International Conference on Machine Learning*, PMLR, 2023, pp. 8469–8488.

[12] Octo Model Team *et al.*, *Octo: An open-source generalist robot policy*, `https://octo-models.github.io`, 2023.

[13] O. X.-E. Collaboration *et al.*, *Open X-Embodiment: Robotic learning datasets and RT-X models*, IEEE International Conference on Robotics and Automation, 2024.

[14] A. Brohan *et al.*, “RT-1: Robotics transformer for real-world control at scale,” *Robotics: Science and Systems (RSS)*, 2023.

[15] B. Zitkovich *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” in *Conference on Robot Learning*, PMLR, 2023, pp. 2165–2183.

[16] M. Kim *et al.*, “Openvla: An open-source vision-language-action model,” *Conference on Robot Learning (CoRL)*, 2024.

[17] A. Khazatsky *et al.*, “Droid: A large-scale in-the-wild robot manipulation dataset,” in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[18] J. Ichnowski *et al.*, “Fogros 2: An adaptive and extensible platform for cloud and fog robotics using ros 2,” in *Proceedings IEEE International Conference on Robotics and Automation*, 2023.

[19] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The google file system,” in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, pp. 29–43.

[20] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, “Towards high performance virtual routers on commodity hardware,” in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008, pp. 1–12.

[21] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, “Turboflow: Information rich flow record generation on commodity switches,” in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–16.

[22] G. Lu, R. Miao, Y. Xiong, and C. Guo, “Using cpu as a traffic co-processing unit in commodity switches,” in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 31–36.

[23] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, “Supporting mobile vr in lte networks: How close are we?” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, pp. 1–31, 2018.

[24] Y. Ren *et al.*, “{Fine-grained} isolation for scalable, dynamic, multi-tenant edge clouds,” in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 927–942.

[25] M. Abderrahim *et al.*, “Efficient resource allocation for multi-tenant monitoring of edge infrastructures,” in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, IEEE, 2019, pp. 158–165.

[26] W. Shao, B. Ye, H. Wang, G. Parmer, and Y. Ren, “Edge-rt: Os support for controlled latency in the multi-tenant, real-time edge,” in *2022 IEEE Real-Time Systems Symposium (RTSS)*, IEEE, 2022, pp. 1–13.

[27] S. Shukla, M. F. Hassan, D. C. Tran, R. Akbar, I. V. Paputungan, and M. K. Khan, “Improving latency in internet-of-things and cloud computing for real-time data transmission: A systematic literature review (slr),” *Cluster Computing*, pp. 1–24, 2023.

[28] L. Shi, I. Ahmad, Y. He, K. Chang, and S. Hussain, “Service group based fofdm-idma platform to support massive connectivity and low latency simultaneously in the uplink iot environment,” *Wirel. Commun. Mob. Comput.*, vol. 2020, Jan. 2020.

[29] H. Yang, A. Alphones, W.-D. Zhong, C. Chen, and X. Xie, “Learning-based energy-efficient resource management by heterogeneous rf/vlc for ultra-reliable low-latency industrial iot networks,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5565–5576, 2020.

[30] H.-C. Yang, T. Bao, and M.-S. Alouini, “Transient performance limits for ultra-reliable low-latency communications over fading channels,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 970–13 973, 2020.

[31] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, “A survey of research on cloud robotics and automation,” *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[32] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC ’12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 13–16.

[33] S. C. Gudi *et al.*, “Fog robotics: An introduction,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[34] N. Tian *et al.*, “A fog robotic system for dynamic visual servoing,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 1982–1988.

[35] S. L. K. C. Gudi, S. Ojha, B. Johnston, J. Clark, and M.-A. Williams, “Fog robotics for efficient, fluent and robust human-robot interaction,” in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, IEEE, 2018, pp. 1–5.

[36] K. E. Chen *et al.*, “FogROS: An adaptive framework for automating fog robotics deployment,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2021, pp. 2035–2042.

[37] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, “Robot operating system 2: Design, architecture, and uses in the wild,” *Science Robotics*, vol. 7, no. 66, eabm6074, 2022.

[38] K. Chen *et al.*, “FogROS2-SGC: A ROS2 cloud robotics platform for secure global connectivity,” *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, 2023.

[39] P. Schafhalter, S. Kalra, L. Xu, J. E. Gonzalez, and I. Stoica, “Leveraging cloud computing to make autonomous vehicles safer,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 5559–5566.

[40] K. Chen *et al.*, “FogROS2-LS: A location-independent fog robotics framework for latency sensitive ROS2 applications,” *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024.

[41] W. Wu, Y. Zheng, K. Chen, X. Wang, and N. Cao, “A visual analytics approach for equipment condition monitoring in smart factories of process industry,” in *2018 IEEE Pacific Visualization Symposium (PacificVis)*, IEEE, 2018, pp. 140–149.

[42] K. Lee and M. Son, “DeepSpotCloud: Leveraging cross-region GPU spot instances for deep learning,” in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Jun. 2017, pp. 98–105.

[43] *GLPK (GNU Linear Programming Kit)*, `https://www.gnu.org/software/glpk/`, Accessed: 2024-09-11.

[44] B. Sredojev, D. Samardzija, and D. Posarac, “Webrtc technology overview and signaling solution design and implementation,” in *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, IEEE, 2015, pp. 1006–1009.

[45] P. Schafhalter, S. Kalra, L. Xu, J. E. Gonzalez, and I. Stoica, *Leveraging Cloud Computing to Make Autonomous Vehicles Safer*, arXiv:2308.03204 [cs], Aug. 2023.

[46] *yolov8_ros For object detection, tracking, instance segmentation and human pose estimation*, `https://github.com/mgonzs13/yolov8_ros`, Accessed: 2024-06-15.

[47] *Discussions on failure to use mutliple interfaces with CycloneDDS*, `https://answers.ros.org/question/375360/multiple-network-interfaces-with-rmw_cyclonedds_cpp/`, Accessed: 2024-07-14.