2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

FogROS2-LS: A Location-Independent Fog Robotics Framework for Latency Sensitive ROS2 Applications

†Kaiyuan Chen[1,2], Michael Wang[2], Marcus Gualtieri[2], Nan Tian[2], Christian Juette[2], Liu Ren[2], Jeffrey Ichnowski[4], John Kubiatowicz[1] and Ken Goldberg[1,3]

***Abstract*— In Cloud Robotics, long system latency due to varying network conditions can cause instability and collisions. However, this can be minimized in the almost univeral case where there are multiple sources available for cloud servers. By extending anycast routing, we introduce FogROS2-Latency-Sensitive, a Fog Robotics framework that offers secure, location-independent connections between robots and latency-sensitive cloud-based servers. FogROS2-LS offloads conventional on-board state estimators and feedback controllers to Cloud and Edge compute hardware without modifying existing applications in ROS2. In the presence of multiple identical services, FogROS2-LS dynamically identifies and transitions to the optimal service deployment that meets latency requirements, thereby empowering robots with limited on-board computing capacity to safely and efficiently navigate dynamic, human-dense environments. We evaluate FogROS2-LS with two latency sensitive case studies: (1) Collision Avoidance: a robot arm guided by visual feedback from consistent distance estimation and collision checking on Cloud and Edge. FogROS2-LS reduces collision failures by up to 8.5x by selecting the best available server, and (2) Target Tracking: FogROS2-LS enables robust and continuous target following and can recover from network failures. Videos and code are available on the website `https://sites.google.com/view/fogros2-ls`.**

I. Introduction

Cloud, or Fog Robotics ([1], [2], [3]) enables robots to access external computing resources for (1) advanced visual perception ([4], [5], [6], [7], [8]); and (2) reinforcement learning-based intelligent motion control ([9], [10]). Our previous work introduced FogROS2—now an official part of the ROS2 ecosystem—which outsources heavy computing tasks to on-demand hardware resources and accelerators, such as GPU, TPU, ASIC, FPGA, and high performance CPU servers.

A common misconception about cloud and fog robotics is that they are unsuitable for latency-sensitive and safety-critical tasks due to network failures and congestion.

In this work, we assume the existence of multiple independent cloud compute servers and providers, and present FogROS2-Latency Sensitive, a Fog Robotics framework that enables reliable latency performances by dynamically selecting the optimal service out of all available servers. We evaluate its effectiveness in real robotics experiments with collision avoidance and continuous target tracking (Fig. 1).

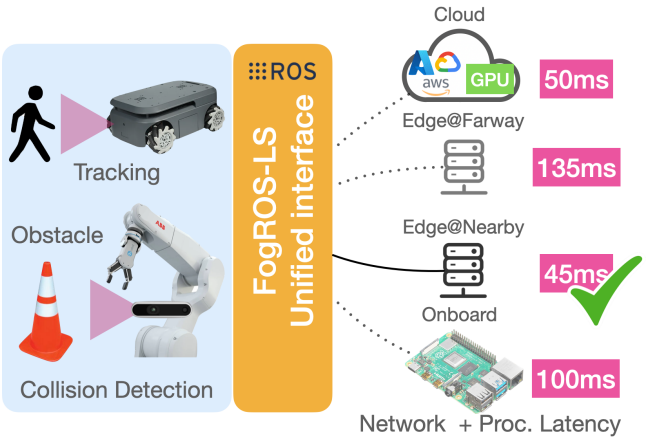


Fig. 1: **A Sample Use Case of FogROS2-LS** FogROS2-LS enables the location-independent deployment of fog robotics applications, allowing robots to connect with distributed robotic services with a unified ROS2 interface. It enables robust operation of latency sensitive applications, such as tracking or collision detection, by connecting robots to service deployments that satisfy these bounds.

ROS2 [11] is the de-facto platform for building robotics applications in a *location-independent* way: heterogeneous robots and modular services[1] publish to and subscribe from (pub/sub) each other as if they are running on the same machine. However, completely adhering to the ROS2 multiple-party pub/sub communication paradigm in fog robotics falls short on the following aspects: (1) Mirrored robotics services can be distributed to heterogeneous geographic locations and network domains. The framework needs to globally and securely discover and connect robots with those service deployments, while differentiating services hosted by other users or tasks. (2) The pub/sub paradigm leads to the request being published to all the deployments, leading to more network congestion and failure. While still adhering to ROS2 interfaces, a robot should select the one deployment that fulfills the application-specific latency bound. (3) Deployment selection should be adaptive to fluctuating application latency caused by varying network latency and hardware resource utilization, and network failures

FogROS2-LS introduces a latency-aware location independent routing architecture. It enables launching multiple instances of robotics task servers across different geographical and network domains, all identifiable by a location-independent identifier unique to the task. Robots use this

[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
[4]Robotics Institute, Carnegie Mellon University
†For correspondence and questions: kych@berkeley.edu

[1]*Service* in this paper refers to generic robotics application instead of the specific ROS2 service communication model. FogROS2-LS supports both publish-subscribe and service communication models in ROS2.

identifier for global discovery and selection of service deployments based on dynamic ROS2 application latency analysis. FogROS2-LS enforces an *anycast* invariant [12], in which each robot subscribes to exactly one service deployment. When failures or latency variations occur, FogROS2-LS seamlessly switches to an alternative deployment that meets the latency constraints, while adhering to the anycast invariance.

To evaluate FogROS2-LS, we performed two latency-critical robotic experiments using FogROS2-LS: (1) collision checking via visual feedback for safe robot arm deceleration, and (2) target following using rapid visual detection feedbacked to a reactive motion planner. Experiments suggest that, compared to using on-board perception modules, Edge/Cloud services via FogROS2-LS provide more timely and reliable performance, enabling safer and more on-track robot control, respectively. We also show that a hybrid service built with FogROS2-LS, a combination of Edge and Cloud service with automatic service switching, is more robust than standard services against system failures because FogROS2-LS allows failure recovery to backup services when one or more services become unreliable or disconnected due to network conditions or computation availability.

This paper claims make four contributions: 1) a location-independent network routing framework that enables robots to discover and connect with fog robotics services across different network domains, 2) a state machine-based Anycast communication paradigm that simplifies the routing management and adheres to the publish/subscribe paradigm in ROS2, 3) adaptive routing based on real-time profiling of latency, and 4) data from an experimental evaluation of FogROS2-LS on two latency sensitive scenarios.

A. Five System features

FogROS2-LS provides the following five key features:

a) No Application Modification: Unmodified ROS2 applications work with FogROS2-LS and operate as though all modules reside on a single computer. FogROS2-LS works for heterogeneous ROS2 transport middleware configurations and network setups.

b) Location Independence: The robotics services can be deployed at different geographic locations and network domains. FogROS2-LS interconnects them with robots as if they were all on the same computer.

c) Latency Boundedness: FogROS2-LS selects and connects to a service deployment that fulfills the time bound defined by the user.

d) Anycast Invariant: FogROS2-LS maintains the anycast invariance that routes robot requests to the optimal service deployment determined by a user-defined latency constraint.

e) Adaptivity: FogROS2-LS adapts its service selection to fluctuating network conditions and application latency.

II. Related Work

A prevailing trend indicates an increasing demand for robots to employ resource-intensive models and algorithms for accurate perception and control. Popular visual perception modules include Segment Anything (SAM) [5], semantic VSLAM [6], Neural Radiance Fields (NeRF) [7], and Language Embedded Radiance Fields (LERF) [8]. Intelligent motion control modules include learning hand-eye coordination with grasping [13], Model Predictive Path Integral (MPPI) [9], and Robotic Transformer 2 (RT2) [14]. The complexity of large or foundational models necessitates more than just the robot onboard computing capabilities. Fog Robotics [1] has been proposed, as a generalization of Cloud Robotics [15], to balance centralized cloud and distributed edge resources to reduce latency, enhance performance, and facilitate real-time processing and decision-making in robotic systems [16]. Fog Robotics have been applied to various robotics applications, such as grasp planning [3], motion planning [2], visual servoing [17], inference [18], and human-robot interaction [19].

Robot Operating System (ROS) 2 [11], the successor of ROS, is the de-facto standard for developing robotics applications due to its broad availability and adaptability. In ROS 2, computational modules are abstracted into *nodes*, and they communicate with each other using a multi-party publish/subscribe paradigm through *topics*. All nodes subscribing to the same topics receive data from other nodes that publish them. Over the past decades, various attempts have been made to enable robotics applications in ROS or ROS2 to leverage Cloud or Fog computational resources. Rapyuta Mohanarajah *et al.* [20] is a proprietary platform for centralized management and deployment of ROS application pipelines. Chen *et al.* [21] propose FogROS, a cloud robotics framework that offloads ROS applications to the public cloud. Ichnowski *et al.* [22] and Chen *et al.* [23] extend FogROS with ten major features, including ROS2 support and major cloud service providers. FogROS2-SGC (Secure and Global Connectivity) [24][25] connects distributed ROS2 nodes through a global peer-to-peer network. In this work, we build upon the FogROS design philosophy, allowing for offloading and connecting robots with ROS2 applications without any code modifications. Furthermore, we extend our study to enabling latency sensitive fog robotics applications. On the technique side, FogROS2 uses Virtual Private Network (VPN) optimized for single cloud and robot, and FogROS2-SGC lacks flexible network routing management. In contrast, our approach in FogROS2-LS leverages the unique communication paradigm Anycast. This allows robots the flexibility to connect to one of several location-independent services and dynamically switch to meet application latency constraints.

III. FogROS2-LS Design

FogROS2-LS, illustrated in Fig. 2, enables latency-sensitive robotics applications by deploying replicated services in a location-independent manner and connecting with user-specified time constraints. Location-independent robotics service is achieved through a secure and global routing framework (Sec. III-A) that associates local ROS2 topics with globally-unique and secure identifiers. The same services can be replicated at different geographic locations with the same identifier, allowing robots to discover the location independent

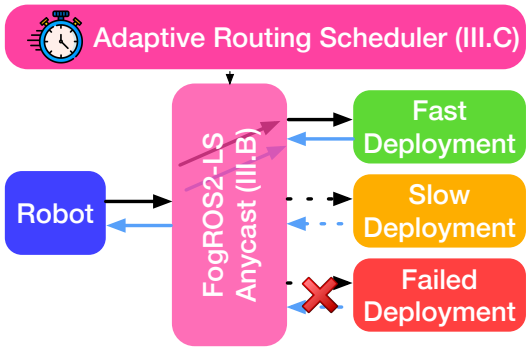


Fig. 2: **System Diagram of FogROS2-LS components** FogROS2-LS enables latency-sensitive fog robotics applications through an adaptive scheduler that generates routing policies. The routing policy directs robots to connect with the optimal service deployment by Anycast.

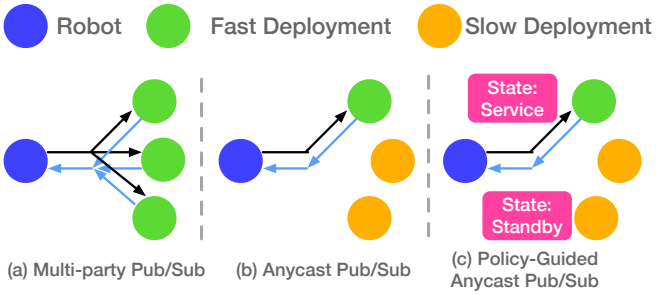


Fig. 3: **Comparison of Publish/Subscribe and Anycast Communication Paradigms** ROS2's publish/subscribe paradigm publishes a message (e.g. a request from robots) to all existing subscribers, which wastes computational and network resources. In our context, the Anycast paradigm is more appropriate, as it forwards to only the most optimal deployment based on the policy.

services and connect as if the service were on the same machine. FogROS2-LS uses Policy-Guided Anycast (Sec. III-B) to address the challenge of choosing a single deployment from several instances sharing the same identifier without any application or interface modifications of ROS2. This approach enables connection to a single service deployment based on a routing policy. This policy is dynamically generated by the Adaptive State Machine Scheduler (Sec. III-C), which monitors and orchestrates application latency.

A. *Location Independent Routing*

Location Independent and Unique Identifiers FogROS2-LS enables multiple location independent deployments of the same service by assigning a shared globally unique identifier to all the service deployments. Robots can deterministically generate the identifier to discover and connect with the service. FogROS2-SGC details the identifier cryptographic construction process, in which an unauthorized attacker attempting a brute-force attack would have to guess the service identifier or reverse the information used to generate it among 10^{77} possibilities, a value near the number of protons in the observable universe. The original FogROS2-SGC uses this property for mobility, allowing robots to move freely across different network domains. FogROS2-LS extends FogROS2-SGC's identifier generation process so that multiple servers at various geographic locations can host the same ROS2 service by assigning their service interfaces with the same identifier. FogROS2-LS also supports identifier generation for both ROS2 publish/subscribe and service/client communication paradigm.

Flexible Global Service Discoverability and Connectivity Given a shared identifier associated with a fog robotic service replicated across various geographical locations, robots face the challenge of globally discovering and connecting to these deployments. This task is challenging due to heterogeneous network domains, firewalls, and Network Address Translation (NAT) [26], which sometimes restrict accessibility to network addresses or ports outside of certain domains. It is further complicated by the flexibility requirement of FogROS2-LS that the robot can dynamically select the service based on the latency requirement.

FogROS2-LS achieves flexible global discoverability by maintaining a *global routing information base*, a centralized registry storing all routing data, enabling newcomers to directly connect with existing publishers and subscribers. This registry solely aids in connection establishment and maintenance without data routing. This provides two clear benefits: (1) it is sufficiently lightweight to operate on low-end servers, with UC Berkeley offering a public version; (2) it enables flexible packet control to specific endpoints by direct robot-service connections, since intermediary routers can introduce protocol complexities and inefficiencies.

FogROS2-LS enables global connectivity through proxying local ROS2 communication to WebRTC [27], a peer-to-peer network transport protocol that facilitates global connections and is commonly used for web-based video conferencing. We refer readers to [27] for its design and guarantees. In FogROS2-LS, the global routing information base maintains the WebRTC channel details of current publishers and subscribers, allowing newcomers to utilize this information to directly establish WebRTC connections with existing participants.

B. *Policy-Guided Anycast*

Many ROS2 applications utilize the publish/subscribe paradigm in ROS2 to emulate a server and client setup, where the robot client publishes to the request topic and subscribes from the response topic, while the server subscribes from the request topic and publishes to the response topic. This emulation works if only one service deployment exists in the network. Otherwise, the robot may send requests to multiple deployments (Fig 3.A), wasting compute and network resources in the context of multiple deployments of the same service. Additionally, the robot could receive duplicated responses from multiple deployments.

Recognizing that robots need to communicate with only one of the services, FogROS2-LS resolves this issue by formulating *Anycast* to describe this *one-to-one-of-many* relationship: the request message only needs to be forwarded to *any* of the deployment that subscribes to the request topic, instead of all of them. Other idle service deployments in ROS2 consume negligible network and CPU resources, which can

be used to handle other robots or services. However, we note that this is conceptually different from IP Anycast [12]: IP Anycast concentrates on routing packets to a destination IP address shared by multiple locations, while FogROS2-LS bridges publish/subscribe to one of many services (Fig 3.B).

FogROS2-LS enables Anycast by managing its global routing state: which ROS2 topics should be globally discoverable, published, or subscribed to. Anycast is achieved when a single service deployment subscribes to service request topics and publishes to service response topics, with the robot maintaining a location-independent connection exclusively with that service deployment. Given that a service may use multiple ROS2 topics, we use *state* to refer to the aggregation of the global topic publish or subscribe relationship of a robot or service deployment at a specific time. An example of the state definition can be found in Listing 1. FogROS2-LS has predefined three states to facilitate Anycast: the *robot* state designated for the robot, the *service* state for actively managing requests, and the *standby* state which prevents the exposure of any global service topics to other machines. One can also define their own state, but additional adaptation is required for anycast invariance.

Each FogROS2-LS-integrated robot or service operates its own state machine. To switch from one service deployment to another, the scheduler reassigns the state machine, marking the old deployment as standby, and the new one as service. FogROS2-LS automates the state transitioning by tearing down the network connections of the previous state, updating the global routing information base, and establishing the connections for the new state.

We transform an intricate network routing management problem into a straightforward state machine management problem. Instead of designing networking protocols and maintaining complex routing states, we only need to keep the invariant that only one service deployment is in *service* state, while other deployments are in *standby* state (Fig 3.C). In FogROS2-LS, the collective states of all the state machines for the robot and service form the *policy*, which is user-initialized (Listing 1) and overseen by an adaptive scheduler (Sec. III-C). To minimize the service interruption of state transitioning, FogROS2-LS maintains the underlying network connection from previous services, merely pausing the packet forwarding instead of completely tearing down the connections. This strategy leverages the minimal overhead involved in keeping a network connection active, allowing for the reuse of these connections when switching for those services that require continuous operation.

C. Adaptive Time-Bounded Policy Scheduler

FogROS2-LS ensures an application's adherence to the time constraint by monitoring the application latency, checking if the latency fulfills the bound, and dynamically adjusting its routing policy. The latency is profiled by adjusting the routing rules ahead of the deployment. The necessity for a user-defined time bound is to avoid impractical monitoring of the most optimal machine and to prevent overly frequent switching.

```
# III.B
state_definition:                                   State Definition
  standby: # do not publish or subscribe to any topics
  service:
    topics:
      - /yolo/input: sub # subscribes to YOLO input
      - /yolo/output: pub # publish to YOLO output
  robot:
    topics:
      - /yolo/input: pub # subscribes to YOLO input
      - /yolo/output: sub # publish to YOLO output
    params:
      - /camera: rgb_module.profile:=640x480x30

                                                    Initial Policy
initial_policy:
  turtlebot: robot
  machine_edge: service
  machine_cloud: standby

# Section III.C Adaptive Latency Monitoring
latency_bound:
  median: 0.3 # in second, max/min/median/mean/stddev
```

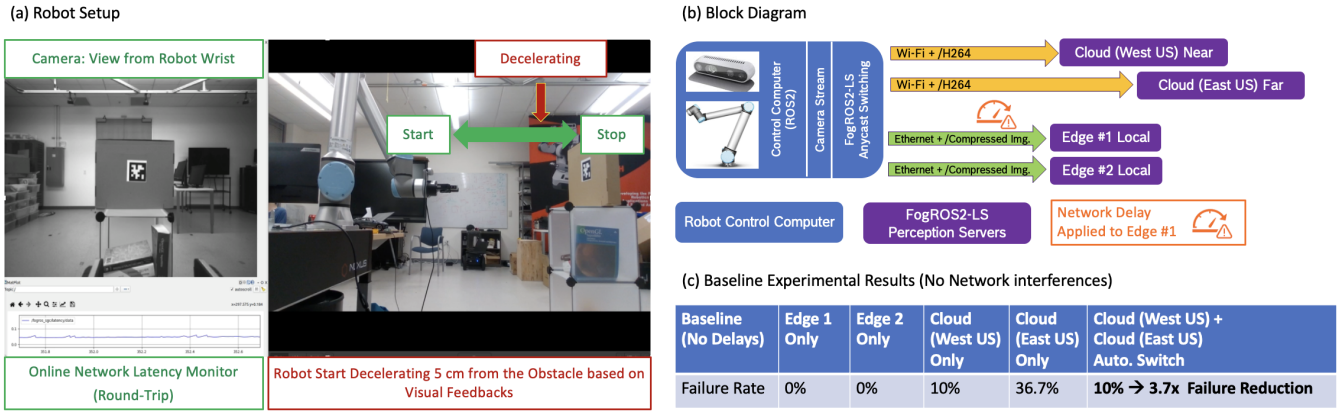
Listing 1: **FogROS2-LS State and Policy Configuration File Example** This `state_definition` section illustrates how the user specifies the machine's state—such as `standby`, which the node remains idle, and `service`, which the node actively listens for input requests and outputs to the response topic. The state machine also allows specifying the parameters such as camera frame rate and resolution. User can also add their own custom states. The second half of the configuration file defines the desired latency bound for the messages from the request topic and response topic.

FogROS2-LS uses a centralized scheduler to determine which machine should be the service machine for the robot. The scheduling decision translates to the actual state machine updates that are synchronized across all available deployments for consistency and fault tolerance. The scheduler passively monitors the application latency and is triggered when the application fails to fulfill the latency bound or if the current service machine is disconnected. The scheduling decision is based on historical profiling results on both active and idle service deployments. The scheduler temporarily directs the messages to the selected service and collects latencies for a short period of time. FogROS2-LS gathers application latency profiles at bootstrapping and when none of the available machines fulfill the latency bound based on the past data.

To accommodate diverse ROS2 applications, FogROS2-LS provides three latency collection mechanisms: (1) measurement of the difference between the request and response time of ROS2 services, (2) heuristics to align the request and response topics in ROS2 pub/sub and to get the timing difference of the request topic and response topic, and (3) direct input of latency from a pre-defined ROS2 interface.

IV. Robotics Evaluation

We demonstrated location-independent FogROS2-LS services with two latency-sensitive physical robotics tasks: (A) high-speed collision avoidance (Fig. 4) and (B) continuous object following (Fig. 5). Both tasks required continuous visual feedback from the robots and time-sensitive controls back to the robots. The first task required a single time-sensitive command while the second required continuous commands. To offload visual perception to both edge and cloud servers, we streamed online camera feeds from these robots to off-board servers via FogROS2-LS for continuous



(c) Baseline Experimental Results (No Network interferences)

Baseline (No Delays)	Edge 1 Only	Edge 2 Only	Cloud (West US) Only	Cloud (East US) Only	Cloud (West US) + Cloud (East US) Auto. Switch
Failure Rate	0%	0%	10%	36.7%	**10% → 3.7x Failure Reduction**

Fig. 4: **High Speed Collision Avoidance Evaluation** (a) The robot arm approached the target with maximum speed and decelerates at 5cm away from the target based on the visual detection of the QR Code. Delayed estimation leads to failure by collision. (b) Architecture of multiple off-board perception servers both on the Edge and in the Cloud (c) Baseline experimental results (without network delays) showing that FogROS2-LS reduces failure by 3.7x when two Cloud servers were available to support a location independent service, compared to the case of a single, far away Cloud (E) service.

Delay added to Edge 1 Failure Rate (%)	Edge 1 Only	Edge 1 + Edge 2	+ Cloud(W)	+ Cloud(E)	+ Cloud(W) + Cloud(E)
x ms Applied to 100% Packets	56.7%	0.0%	10.0%	30.0%	**6.7%**
(Failure Reduction)		(∞)	(5.7x)	(1.9x)	**(8.5x)**
x ms Applied to 50% Packets	26.7%	0.0%	**6.7%**	53.3%	10.0%
(Failure Reduction)		(∞)	**(4.0x)**	(0.5x)	(2.7x)

TABLE I: **When local network was congested, FogROS2-LS reduced failures:** Upon application of random network delays on Edge #1, FogROS2-LS automatically re-route to alternative services, i.e. Edge 2, Cloud(W), Cloud(E). Such automatic switching functionality to an alternative, location independent service(s) improves robustness of the system and reduced failure rate (lower the better) of the overall system. FogROS2-LS matches performance of the best individual machine, reducing the failure rate by up to 8.5 times by selecting the best available machine compared to single edge service system under heavy network congestion. We also showed that FogROS2-LS enabled flexible yet reliable access to open cloud services via Anycast to reduce failure rate.

QR Code pose estimation (AprilTag [28]). 6D pose of the target and robotics control signals were then sent back, also via FogROS2-LS, to complete the feedback loop. FogROS2-LS continuously monitored the round trip time of each frame of pose estimation on all available edge and cloud servers connected to the operating robot during run time.

When the current service failed to return 6D poses to the robot within the designated time, FogROS2-LS autonomously switched to another operational service. This approach enhances system robustness, both by averting potential failures, such as in collision avoidance scenarios, and by facilitating recovery in instances of continuous object tracking.

A. *Collision Avoidance*

Setup Fig. 4 shows the physical setup of the collision avoidance evaluation. The Universal Robots arm (UR10e), with an Intel RealSense D435i mounted on the wrist, advanced towards the QR Code at maximum speed (measured at 1.3 m/sec). Meanwhile, the camera sent a monocular video stream to the Edge or Cloud at 640x480 resolution, maintaining a steady QR Code 6D pose estimation at 90 Hz. Due to bandwidth limitations, we streamed the video to the cloud with H.264 compression. We streamed Video to a local edge server via series of compressed images rather than H.264, as edge network has better bandwidth. The robot arm underwent maximum deceleration if robot arm recognized that it was 5 cm away from the target. It is considered failure if the off-board robot command, in response to the visual detection, failed to reach the robot in time, causing the robot arm to collide with the target that holds the QR code.

We used FogROS2-LS to connect the robot controller to various Edge or Cloud servers. We isolated the ROS nodes on the robot controller from Edge server's using ROS2 domains to avoid cross interference. All the experiments were conducted 30 times. Prior to each run, FogROS2-LS profiled networks for up to 100 ms per machine. It then selected the machine with the lowest round-trip latency that can fullfill the configured time-bound. It then pauses for 500 ms to allow H.264 stabilization before each robotics trial begin.

Results To benchmark, we calculated the failures rate over the 30 trials for baseline (Fig. 4 (c)) and local network congestion (Table I). We showed that the location-independent service built with FogROS2-LS reduced the failure rate of the collision avoidance robot task, because FogROS2-LS can choose the best available service automatically via Anycast.

FogROS2-LS can reduce the failure rate by up to 8.5 times. When 100% network latency is applied and the cloud is available, FogROS2-LS chooses Cloud for almost all the trials, and the overall performance is close to the performance of the Cloud. Notably, FogROS2-LS can perform better than any available standalone machines by selecting the available machine with best latency at a given moment. It can achieve an even lower failure rate of up to 1.5 times lower than that of any single machine available.

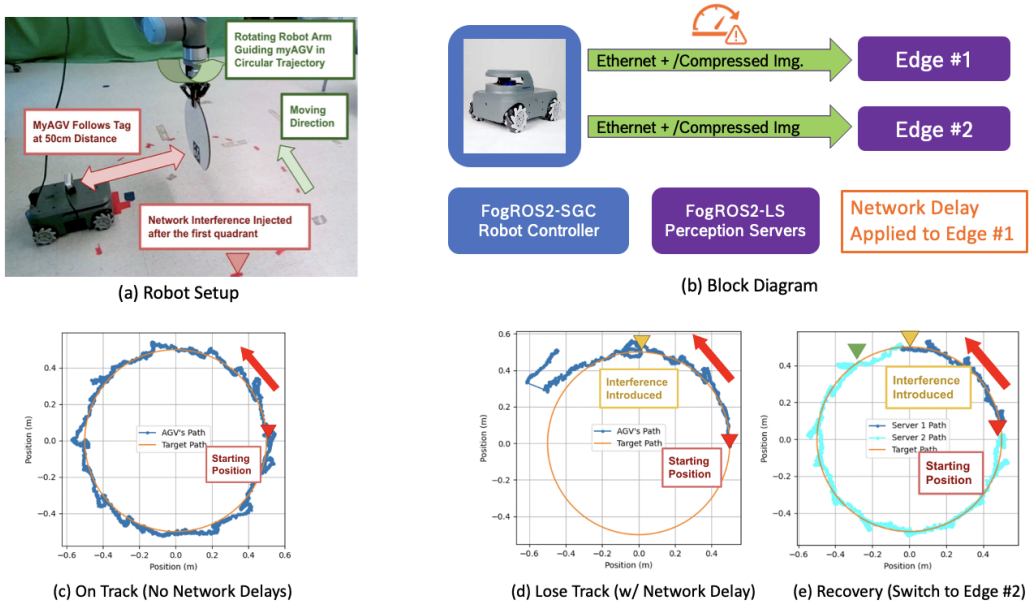


Fig. 5: **Position of Mobile Robot Relative to a Circular Path**. (a) & (b) Setup & Architecture; (c), the mobile robot successfully follows the circular path when no network inference was introduced; (d), a 50 ms network delay was introduced after the initial quadrant, causing mobile robot to immediately deviate from the path; (e) showcases FogROS2-LS's ability to recover from the failure encountered by transitioning to another Edge server.

In all the series of experiments, one failure case occurred– when 50% latency was applied to Edge 1 and FogROS2-LS offload perception to Cloud US East. The failure rate of the combined Cloud-Edge was worse than that of the original Edge 1 machine. This outcome was caused by imprecise network profiling due to the following reasons: (1) substantial network variation for Edge and Cloud is not practical for optimization; (2) while the FogROS2-LS network profiler operates under the assumption of uncorrelated request latencies, the H.264 compression alternates between transmitting a complete image frame and sending only the difference from the previous frame, so request latencies can be correlated in time.

B. Continuous Target Following

Setup We used MyAGV [29] for a continuous target following experiment. This fast-moving mobile vehicle has Mecanum omnidirectional wheels, a Raspberry Pi 4, and an Intel RealSense D435i camera. During the experiment, it tracked the QR code on the UR10e robot arm such that the robot aimed to stay 0.5 meters away in the normal direction of the tag. The wrist of UR10e maintains a rotation speed of 0.075 rad/s, guiding mobile robot in a circular path. In the meantime, the Raspberry Pi runs FogROS2-LS to stream the RGB video to the Edge server at 424x240 resolution at 30 Hz with an Ethernet cable. The position of MyAGV w.r.t. the start is estimated as $R(\theta)x$, where $R(\theta)$ is a 2x2 rotation matrix with angle θ, θ is the angle reading of the UR10's wrist joint, and x is the detected 2x1 position of MyAGV's camera w.r.t. the QR code. In the evaluation, we use two identical Edge servers connecting to MyAGV via FogROS2-LS, one executes QR Code detection, while a backup Edge server remains on standby. While the MyAGV progresses past the first quadrant (90 degrees) of the target circluar path, we introduce a 50ms network latency, causing a tracking failure and deviation from the trajectory. A successful FogROS2-LS switch should facilitate tracking recovery, allowing the system to resume object tracking by leveraging the backup Edge server.

Results Fig. 5 shows the the results of target following with FogROS2-LS. In the presence of 50 ms network latency after the initial quadrant, MyAGV deviates from the path. FogROS2-LS is able to recover from the failure encountered in (e) by transitioning to another Edge server.

V. Conclusion and Future Work

This work introduces FogROS2-LS a location-independent ROS2 Framework for latency-sensitive cloud robotics. One limitation is that the accuracy of latency profiling is affected by the correlation of the message processing latencies. For example, H.264 video compression can result in imprecise latency estimations where some packets represent complete frames and others contain only the differences from the preceding frame. In future work, we will model the predictability of such packet dependency for robotics applications.

VI. Acknowledgement

The authors would like to thank Dirk Elias, Philipp Mundhenk, Arne Hamann, Ralph Lange from Bosch Corporate Research, Renningen, for discussions on Reliable Distributed Systems (RDS) with Cloud. We also thank Ajay Tanwani from Bosch for feedback. This work is funded and supported by the large scale collaboration (LSC) between UC Berkeley and Robert Bosch Research and Technology Center, North America. Kaiyuan Chen and John Kubiatowicz are also funded by C3.AI.

REFERENCES

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

[2] 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.

[3] 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.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[5] A. Kirillov *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[6] C. Yu *et al.*, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2018, pp. 1168–1174.

[7] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "Nerf: Neural radiance fields without known camera parameters," *arXiv preprint arXiv:2102.07064*, 2021.

[8] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," *arXiv preprint arXiv:2303.09553*, 2023.

[9] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 1433–1440.

[10] A. Brohan *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[11] 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.

[12] H. Ballani and P. Francis, "Towards a global ip anycast service," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 301–312, 2005.

[13] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[14] B. Zitkovich *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of The 7th Conference on Robot Learning*, J. Tan, M. Toussaint, and K. Darvish, Eds., ser. Proceedings of Machine Learning Research, vol. 229, PMLR, Jun. 2023, pp. 2165–2183.

[15] 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.

[16] S. Chinchali *et al.*, "Network offloading policies for cloud robotics: A learning-based approach. arxiv e-prints, page," *arXiv preprint arXiv:1902.05703*, 2019.

[17] 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.

[18] A. K. Tanwani, R. Anand, J. E. Gonzalez, and K. Goldberg, "RILaaS: Robot inference and learning as a service," *IEEE Robotics & Automation Letters*, vol. 5, no. 3, pp. 4423–4430, 2020.

[19] 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.

[20] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2014.

[21] 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.

[22] J. Ichnowski *et al.*, "FogROS2: An adaptive and extensible platform for cloud and fog robotics using ros 2," *arXiv preprint arXiv:2205.09778*, 2022.

[23] K. Chen *et al.*, "Fogros2-config: A toolkit for choosing server configuration for cloud robotics," *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024.

[24] K. Chen, J. Yuan, N. Jha, J. Ichnowski, J. Kubiatowicz, and K. Goldberg, "FogROS G: Enabling secure, connected and mobile fog robotics with global addressability," *arXiv preprint arXiv:2210.11691*, 2022.

[25] K. Chen *et al.*, "FogROS2-SGC: A ROS2 Cloud Robotics Platform for Secure Global Connectivity," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 2035–2042.

[26] G. Tsirtsis and P. Srisuresh, "Network address translation-protocol translation (nat-pt)," Tech. Rep., 2000.

[27] 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.

[28] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 4193–4198.

[29] *My AGV Mobile Robot*, `https://shop.elephantrobotics.com/products/myagv`.