

FogROS2-SGC: A ROS2 Cloud Robotics Platform for Secure Global Connectivity

Kaiyuan Chen¹, Ryan Hoque¹, Karthik Dharmarajan¹, Edith LLontop¹, Simeon Adebola¹, Jeffrey Ichnowski², John Kubiatowicz¹, and Ken Goldberg^{1,3}

Abstract—The Robot Operating System (ROS2) is the most widely used software platform for building robotics applications. FogROS2 extends ROS2 to allow robots to access cloud computing on demand. However, ROS2 and FogROS2 assume that all robots are locally connected and that each robot has full access and control of the other robots. With applications like distributed multi-robot systems, remote robot control, and mobile robots, robotics increasingly involves the global Internet and complex trust management. Existing approaches for connecting disjoint ROS2 networks lack key features such as security, compatibility, efficiency, and ease of use. We introduce FogROS2-SGC, an extension of FogROS2 that can effectively connect robot systems across different physical locations, networks, and Data Distribution Services (DDS). With globally unique and location-independent identifiers, FogROS2-SGC securely and efficiently routes data between robotics components around the globe. FogROS2-SGC is agnostic to the ROS2 distribution and configuration, is compatible with non-ROS2 software, and seamlessly extends existing ROS2 applications without any code modification. Experiments suggest FogROS2-SGC is 19× faster than rosbridge (a ROS2 package with comparable features, but lacking security). We also apply FogROS2-SGC to 4 robots and compute nodes that are 3600 km apart. Videos and code are available on the project website <https://sites.google.com/view/fogros2-sgc>.

I. INTRODUCTION

As robots are increasingly deployed worldwide, they require mechanisms to efficiently, reliably, and securely communicate with other robots, sensors, computers, and the cloud. The applications are broad, from mobile robots with changing IP addresses due to traveling through different networks, to a fleet of globally distributed robots learning collaboratively. Cloud and fog robotics [1] empower robots and automation systems to harness off-board resources in cloud-based computers. In prior work, we introduced FogROS2 [2], now an official part of the ROS2 ecosystem [3], to enable robots to execute modern compute and memory-intensive algorithms using on-demand hardware resources on the edge and cloud. However, robots connecting to the cloud, a nearby computer on a different network, or a robot halfway around the world introduce additional challenges: (1) Making robots accessible to other robots on the public internet may leave them vulnerable to unauthorized connections and data breaches. (2) The heterogeneity of interconnected devices, communication protocols, and configurations causes

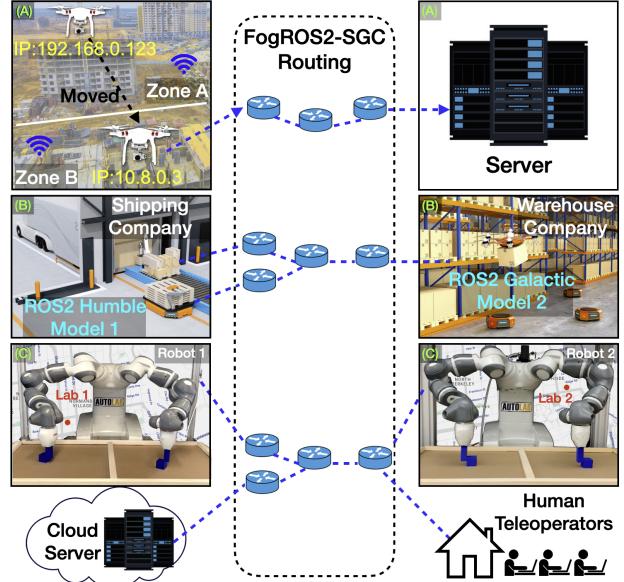


Fig. 1: FogROS2-SGC enables Secure Global Connectivity for robots, allowing robots to communicate with other robots, computers, and the cloud through a familiar ROS2 interface. With FogROS2-SGC, (A) drones navigating large construction sites can seamlessly communicate, even when their IP addresses are constantly changing due to switching Wi-Fi and cellular networks; (B) shipping and stocking robots from different corporations can securely share only the required topics necessary to facilitate the transfer of goods at a warehouse; and (C) globally distributed robots can participate in fleet learning. In experiments, we demonstrate FogROS2-SGC on Fleet-Dagger [4], a fleet learning algorithm, with 4 robot arms operating simultaneously in different locations.

incompatibilities that hinder integration and operation. (3) The changing network topology of mobile robots and Unmanned Aerial Vehicles (UAVs) challenges their ability to stay connected. To illustrate some of these challenges (Fig. 1), consider:

(A) *Security and inspection drones*: Drones navigate a construction site and stream data to a central station that updates a dynamically-changing SLAM map [5]. As drones fly through different cellular and Wi-Fi networks, their IP addresses change, but they should remain securely connected.

(B) *Coordinating heterogeneous mobile robots in a warehouse*: Robots belonging to different companies (e.g., shipping vs. warehouse) and of different makes and models hand off items between container and warehouse. Each robot has unique software packages and versions (e.g., operating systems and network protocols) and must communicate, but only a few selected topics are necessary for the handoff.

(C) *Distributed fleet learning*: Robot arms at different locations pool their data and collectively update a shared

¹Department of Electrical Engineering and Computer Science

²Carnegie Mellon University, Pittsburgh, PA, USA

³Department of Industrial Engineering and Operations Research

^{1,3}University of California, Berkeley, CA, USA

kych@berkeley.edu

control policy. Robots unable to make progress can fall back on remote human teleoperators, using algorithms such as Fleet-Dagger [4].

To address these challenges, we present FogROS2-SGC (Secure Global Connectivity), an extension of FogROS2 that securely and reliably connects robots across different software components, network protocols, and physical locations. FogROS2-SGC enables disjoint ROS2 networks to connect to ROS2 topic interfaces named with *globally-unique* and *location-independent* identifiers. The robots using FogROS2-SGC can roam freely while staying connected because the identifiers are constant. The identifiers are 256-bit strings that are secure by construction—a brute-force attack would have to find a match among 10^{77} possibilities (a value close to the number of protons in the observable universe¹). FogROS2-SGC adopts a *security-first* routing design, where only authenticated parties can connect to the robot and establish secure communication. In contrast to prior work such as SROS2 [7] and FogROS2 [2], FogROS2-SGC does not require merging distributed ROS2 networks, allowing robots to keep their ROS2 networks private and expose public topics only if explicitly configured. Providing fine-grain isolation and access control reduces the attack surface and enhances scalability.

FogROS2-SGC seamlessly integrates with ROS2 applications without code modifications via an SGC proxy. Its implementation and security policy configuration are agnostic to ROS2 distributions and their network transport middleware vendors. FogROS2-SGC is also compatible with non-ROS2 programs that interact with ROS2 components and can provide secure global connectivity to non-cloud servers and computers. Furthermore, since memory copy and synchronization operations are expensive for memory-constrained robots, the implementation of FogROS2-SGC processes can route data without performing unnecessary copies (also known as “zero copy”).

Experiments suggest that FogROS2-SGC reduces the network latency of a cloud-based grasp planning application by $9.42\times$ compared to unsecured rosclt [8]-rosbridge [9]. We also deploy FogROS2-SGC to simultaneously control a fleet of 4 robot arms in different physical locations with compute off-loaded to a server 3600 km away.

This paper makes the following contributions: (1) FogROS2-SGC, an extension of FogROS2 that connects disjoint ROS2 networks by assigning public ROS2 topics with globally-unique and location-independent identifiers. (2) Method for secure and efficient routing with FogROS2-SGC. (3) A Rust implementation of FogROS2-SGC that uses zero-copy message processing and asynchronous network operations for robots with memory and compute constraints. (4) Evaluation of FogROS2-SGC on cloud robotics applications (vSLAM, grasp planning, motion planning, simultaneous fleet control) demonstrating up to $9.42\times$ latency reduction and enhanced usability.

	(c) security	(e) isolation	(f) global connectivity	(g) resilient connectivity	(h) DDS vendor agnostic	(i) non-ROS compatibility	(j) efficient message processing
SROS2/Cyclone	✓	✗	—	✗	✗	✗	✓
SROS2/FastDDS	✓	✗	—	✗	✗	✗	✓
rti_connect	—	—	—	—	✗	✗	✓
Rosbridge	✗	✓	✗ *	✗ *	✓	✓	✗
Zenoh	—	✗	✓	✓	✗	—	✓
FogROS2	✓	✗	—	✗ *	—	✗	✓
FogROS2-SGC	✓	✓	✓	✓	✓	✓	✓

✗ Not supported — Not trivial ✓ Supported

Fig. 2: **Comparison of FogROS2-SGC with other distributed ROS2 systems.** In this table, we compare the feature support of different distributed ROS2 systems with the features in Section III. Some features can be supported but require non-trivial effort beyond changing the configuration. For example, both the routing service in rti_connect and discovery server in FastDDS/SROS2 support global connectivity but require manually modifying routing rules or setting up a point-to-point VPN when a new node joins [10]. Rosbridge and FogROS2 support only unidirectional global and resilient connectivity (marked with *), meaning that one side of the communication must have a fixed IP. In contrast, the identifier-based routing of FogROS2-SGC allows either side to have a dynamic IP address.

II. RELATED WORK

James Kaufner introduced the term ‘Cloud Robotics’ in 2010 [1]. Cloud and fog computing have been applied to robotic tasks such as grasp planning (Tian et al. [11], Kehoe et al. [12], and Li et al. [13]), parallelized Monte-Carlo grasp perturbation sampling (Kehoe et al. [14], [15], [16]), and motion planning (Lam et al. [17]). Chen et al. [18] and Ichnowski et al. [2] propose frameworks for offloading computation to resources on the edge or cloud, while Ichnowski et al. [19] and Anand et al. [20] present systems that leverage serverless computing [21]. Modern computing paradigms have enabled new applications such as multi-robot interactive fleet learning (Swamy et al. [22], Hoque et al. [4]) and remote sharing of robot systems (Tanwani et al. [23], Bauer et al. [24]).

Remote interactions between robots and the cloud raise security, compatibility, and connectivity challenges for robots. Virtual Private Networks (VPNs) are the most common approach for establishing secure communication between robots and the cloud for both ROS and ROS2 (e.g., Lim et al. [25]). Establishing a VPN link between a robot and the cloud is a complex process [26]. FogROS [18] and FogROS2 [2] automate the certificate generation and VPN setup. SROS2 [7] is an alternative approach to securing ROS2 communication that enforces access control of ROS2 topics. However, it requires DDS-dependent discovery mechanisms to ensure connectivity. Discovery mechanisms for DDS (such as the discovery server for FastDDS [27] and the RTI routing service for RTI Connect [28]) are vendor-specific and not compatible with other DDS implementations. Zenoh for ROS2 [29] is integrated with CycloneDDS to enhance peer-to-peer connectivity, but it is not compatible with other DDS implementations. ROS Remote [30] by Pereira et al. and MSA [31] by Xu et al. propose alternative protocols

¹The Eddington Number [6] (N_{Edd}) is currently estimated to be 10^{80} .

to unify cloud-robot communication. However, alternative protocols require modifications to ROS applications and are not compatible with ROS2. Finally, rosbridge [9] proposed by Crick et al. is widely adopted by both ROS1 and ROS2 to allow non-ROS software to interact with ROS2 nodes. It can also be used to bridge two non-compatible and remote ROS applications when used in conjunction with rosduct [8]. However, rosduct and rosbridge have significantly high message latency when the message size is large (e.g., images). A summary of how FogROS2-SGC differs from related work can be found in Fig. 2.

III. TEN FOGROS2-SGC FEATURES

FogROS2-SGC addresses the Secure Global Connectivity (SGC) problem of securely and reliably connecting globally distributed robots, sensors, computers, and the cloud by extending FogROS2. We enumerate 10 new features to differentiate from related libraries and alternative approaches.

a) Globally identifiable addresses: FogROS2-SGC enables a scalable number of ROS2 networks to publish a subset of ROS2 topics to other disjoint ROS2 networks around the globe. In scenario (C) from Fig. 1, the robot arms are located at different geographic locations with different local ROS2 networks. FogROS2-SGC allows remote human teleoperators to operate the robot arms as if the arms are connected to local networks. FogROS2-SGC also allows disjoint robots to publish to the same local ROS2 topics with globally unique and identifiable addresses.

b) Transparency to ROS2 applications: ROS2 modularizes a robotics application into *nodes*, and connects the nodes into a graph. Nodes communicate with each other through a publish-subscribe (pub/sub) system, where publisher nodes send messages to *topics*, and nodes subscribed to these topics receive these messages. FogROS2-SGC adheres to the abstractions and interfaces of ROS2. ROS2 applications interact with remote nodes as if they are nodes on the same robot or subnetwork.

c) Communication security: FogROS2-SGC guarantees that no unauthorized attacker can eavesdrop or tamper with ROS2 messages. Authorization is identified by user-configured cryptographic keys. In all three scenarios from Section I, the communication goes through wide-area network with untrusted infrastructure. FogROS2-SGC prevents attackers from accessing any content in ROS2 messages and differentiates authentic robots from spoofing attackers.

d) Global anonymity: Authorized participants can deterministically derive global identifiable addresses with ROS2 topic information and cryptographic secrets. Attackers cannot reverse any information used to recover addresses or topics. FogROS2-SGC guards against attackers knowing part of the ROS2 topic information deducing the global address. For example, the attackers who know the topic name and type information cannot guess the address, because they miss the author information and security credentials of ROS2 node.

e) ROS2 network isolation and topic-level access control: FogROS2-SGC connects robots without merging distributed ROS2 networks. Every robot can have an arbitrary

number of private ROS2 topics and only public interfaces are shared with other authorized ROS2 networks. Other ROS2 nodes interact with these public interfaces just as they interact with a local ROS2 topic. This protects the privacy of the robot and prevents unintended messages from being shared with other disjoint networks of the system. For example, in scenario (B), a delivering robot from one company and receiving robot from another may have some proprietary topics that are kept private from each other. FogROS2-SGC isolates the topics private to the robots.

f) Global connectivity: Some robots are connected to subnetworks that are not directly accessible from the outside. For example, robots in scenario (C) are in local area networks behind Network Address Translation (NAT). NAT allows multiple robots to share the same IP, but the translation is dynamic and ROS2 nodes outside cannot directly access the robots. FogROS2-SGC can connect ROS2 nodes that are behind firewalls and NAT.

g) Seamless and resilient connectivity to network dynamism: FogROS2-SGC adapts to the dynamic network behaviors of drones and mobile robots. FogROS2-SGC does not rely on static IP addresses to identify the robots because such addresses are usually bound to a physical location. Adding or reconnecting to robots should not restart the ROS2 node entirely, as this causes service interruptions and failures.

h) DDS-agnostic compatibility: ROS2 adopts the Data Distribution Service (DDS) as its underlying network transport middleware to marshal, unmarshal, and exchange messages. ROS2 supports different DDS implementations, such as CycloneDDS [32], FastDDS [27], and RTI Connex [28]. However, a warehouse in scenario (B) may have robots running different versions of ROS2 and DDS. FogROS2-SGC is DDS-agnostic by leveraging ROS2 abstractions and not using any DDS-specific interfaces.

i) Compatibility with non-ROS2 software: FogROS2-SGC allows non-ROS2 software to interact with ROS2 nodes. This principle is inspired by rosbridge [9]. Besides common transport protocols, while ROS2 officially supports C++ and Python, FogROS2-SGC allows programs to use gRPC [33], the most widely used Remote Procedure Call (RPC) framework that can run in heterogeneous environments and major popular programming languages (such as Go, Java, Javascript, PHP, Rust), to control the robots.

j) Efficient message processing and routing: ROS2 messages are buffered in memory to be processed by FogROS2-SGC. Because robots often have memory and compute resource constraints, FogROS2-SGC is memory-efficient by reducing unnecessary message copying and memory synchronization. Since FogROS2-SGC requires frequent exchange of packets to and from the network, network operations (such as `send` and `recv`) are not on the critical path of message processing.

IV. FOGROS2-SGC DESIGN

See (Fig. 3) for system architecture. FogROS2-SGC sends messages via a globally unique identifier (Sec. IV-A). This identifier is unique to a robot and topic pair; thus, it can

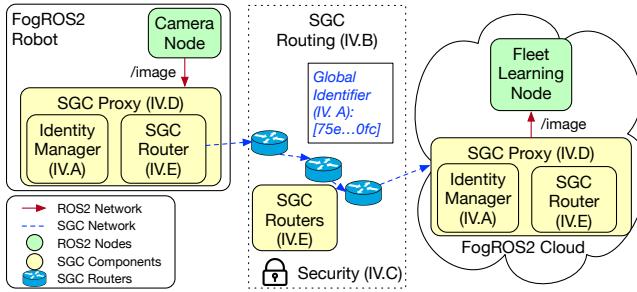


Fig. 3: System overview of FogROS2-SGC’s architecture showing a connection between a robot camera stream (on the ROS2 topic `/image`) and the cloud. The FogROS2-SGC assigns the ROS2 topic `/image` an anonymous, globally-unique and location-independent 256-bit identifier `[75e...0fc]` (truncated for brevity). The messages between identifiers are securely routed with SGC router.

be used for sending and receiving messages regardless of robot location or network address (Sec. IV-B). The identifier is secure, and communication is encrypted, meaning only authorized robots and nodes can access messages from its referenced topic (Sec. IV-C). To implement routing based on the identifier, FogROS2-SGC consists of two main software components—(1) a router (Sec. IV-B and IV-C), responsible for securely routing messages between other routers and nodes, and (2) a proxy (Sec IV-D), that converts between ROS2 messages and the secure routers. As robots can be compute- and memory-constrained, FogROS2-SGC provides an efficient implementation (Sec. IV-E).

A. Global Addressability

Maintaining a globally unique identifier enables the identification of a specific robotic component across subnetworks. FogROS2-SGC uses ROS2 topics as the minimal granularity for the global identifier because a topic is an interface to ROS2 nodes, and a ROS2 node can publish or subscribe to multiple ROS2 topics at the same time. For example, a ROS2 vSLAM node in openVSLAM [5] has four ROS2 topics for camera information, video streaming, output localization, and mapping information. These ROS2 topics expose standardized interfaces with fixed message types. Users can limit the exposure of the ROS2 network by allowing only parts of the interface to be public. Partitioning public and private interfaces also enhances privacy and isolation, prevents unintended message exchanges, and reduces communication overhead.

The identifier is designed to be unique, deterministic, and location-independent. To avoid name collisions, every identifier has 256 binary bits, leading to 2^{256} possible identifiers. Instead of letting users decide, all identifiers are cryptographically derived from the metadata of the ROS2 topics by an identifier manager in SGC proxy. The SGC proxy collects metadata such as the ROS2 node’s name, author, maintainer, interface, and description from standard ROS2 interface and user configuration file. The metadata also has a unique string in case the user needs to deploy the same topic at different locations. Every topic has an associated security certificate in X.509 [34] to verify the identity of those who want to publish or subscribe to the network. All

the metadata is serialized and converted into a 256-bit string using SHA-256 [35], a widely used cryptographic hashing algorithm that maps arbitrary lengths of text to almost-unique 256-bit binary strings.

Security Analysis: The hashed string is suitable for use as the globally unique identifier for the following reasons: (1)

Deterministic: The hash is deterministic so that every party holding the same metadata can derive the same hash value and thus the same global identifier. (2) **One-way:** SHA-256 is a one-way function, so the attacker cannot deduce or reverse the original metadata from the 256-bit identifier. (3) **Avalanche effect:** A small change to the original metadata leads to a new hash value that appears unrelated to the original hash value.

(4) **Large namespace:** There are 2^{256} possible identifiers and it has been proved to be computationally intractable to find two messages with the same hash. Verification of these guarantees can be found in Appel [36].

B. Location-Independent Routing

Although having all identifiers in the same globally-flat namespace protects the privacy of the node’s identity information and physical location, the identifiers do not carry any routing information. Flipping a bit in the identifier may lead to a drastic change in its physical location, or from existent to nonexistent. Therefore, securely routing messages between flat identifiers is a challenging problem. To solve this problem, FogROS2-SGC consolidates and extends the Global Data Plane (GDP) [37], a peer-to-peer network that routes messages between location-independent identifiers. The routers are set up by the user and peer-wise connected into a routing graph; robots do not need to know other robots’ addresses as long as there is a connected routing path. The routers can be any machine that has network and general compute capabilities, such as an edge computer or a cloud server. Every router stores the mapping between the identifiers and the corresponding routing information of the identifiers in Routing Information Base (RIB).

A joining robot or router broadcasts an *advertisement* packet that announces the existence of the identifier and the routing information to the robot. The packet format is aligned with other FogROS2-SGC packets in Fig. 5. Other routers store the routing information in RIB and broadcast the advertisement packet. Routing is achieved by looking up the destination routing information in the RIB and forwarding to that destination.

Fig. 4 illustrates a step-by-step example of a publishing and subscribing `/camera` topic with FogROS2-SGC. The figure assumes that all the connections between routers are established. This can be achieved through configuration or dynamic node discovery [38]. The steps are: (1) The robot SGC proxy P1 generates an advertisement message for the ROS2 topic `/camera` and sends it to Router 1. (2) After verifying the advertisement message, Router 1 records the advertisement in its RIB and forwards the topic information to Router 2. There can be multiple routers between Router 1 and Router 2. (3) Cloud SGC proxy P2 requests to subscribe to `/camera`, and the subscribe request is sent to Router 2.

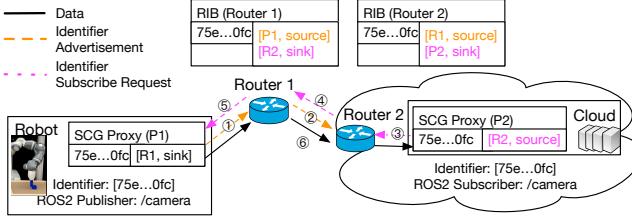


Fig. 4: An illustration of how a routing connection is established between robot and cloud. The steps are further described in Section IV-B. (1,2) Advertisement generation and publish. (3,4,5) Subscribe request. (6) Data routing.

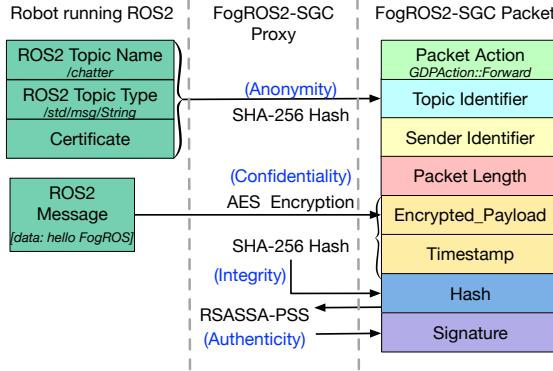


Fig. 5: An illustration of the cryptographic tools used by SGC proxy to protect a ROS2 string message. The FogROS2-SGC Packet on the right is the message that is routed by FogROS2-SGC. The payload is encrypted to protect the confidentiality of the original ROS2 message. The encrypted data is hashed so that the receiver can verify the message is intact. The hash is signed with the sender's key so that the receiver can verify that the message comes from an authentic and authorized sender.

(4) The subscribe request from Router 2 is routed to Router 1 by checking the source information at Router 2's RIB. After verifying the request, Router 1's RIB records P2 as the data sink. (5) The subscribe request from Router 1 is routed to the robot by checking the source information at Router 1's RIB. If the destination is not found, the router broadcasts a query to other routers. (6) The robot's ROS2 publisher sends a ROS2 message to the proxy. The proxy forwards it to Router 1, Router 1 forwards to Router 2, and Router 2 to the cloud subscriber. At each hop, the messages are forwarded from source to sink.

C. Secure Communication

The security of the communication is achieved by using a secure network protocol between routers. We use Datagram Transport Layer Security (DTLS) [35] to provide communications privacy. The DTLS protocol provides secure and authenticated communication on User Datagram Protocol (UDP) and includes a built-in mechanism for dealing with lost or out-of-order packets. DTLS on UDP is well suited for latency-critical robotics communications systems, due to its lightweight nature and low overhead compared to transmission Control Protocol(TCP). The cryptographic algorithms used to secure ROS2 packet generation process can be found in Fig. 5. The message has the following security guarantees: **Confidentiality:** The ROS2 messages are encrypted with AES Encryption [39] to ensure that only parties with the correct cryptographic key can decrypt the original ROS2

message data. **Integrity:** The encrypted message is hashed by SHA-256 [36] so the receiver or third-party auditor can easily verify that the message is intact and no other attacker has tampered with the message. **Authenticity:** The hashed message is signed by the RSASSA-PSS [40] algorithm so that receivers can verify that the message is sent from an authorized sender.

To tailor the security with the communication patterns of robotics applications, FogROS2-SGC allows flexible peering with other routers or end points. One may choose to use a dedicated DTLS connection per ROS2 topic, which is ideal for large message payload and frequent communication (e.g., video streaming). One may also choose to use a shared DTLS tunnel, where multiple ROS2 topics share the same DTLS connection. Sharing the same connection reduces the cost of secure connection management and message processing, which is good for small message payloads and less frequent communication.

D. Transparent and Compatible SGC proxy

SGC proxy is the interface between FogROS2-SGC and the ROS2 network. In order to allow seamless integration with **any** unmodified ROS 2 application code and mainstream DDS vendors, SGC proxy converts between ROS2 communication and FogROS2-SGC communication bidirectionally. The user first identifies ROS2 topics that they wish to publish or subscribe through a configuration file. The proxy launches a local ROS2 publisher or subscriber for the corresponding topic. New messages from the local ROS2 network are actively subscribed to by the proxy, and sent to the FogROS2-SGC network. Once the verified subscribers receive the messages, they convert them to standard ROS messages and publish to their local ROS2 network.

To allow non-ROS2 programs to communicate with ROS2 nodes, SGC proxy converts ROS2 messages to a unified JSON-based message format in transit. As a result, FogROS2-SGC can be extended to a variety of protocols such as TCP, UDP, DTLS, TLS, and gRPC. Note, however, that some of the protocols need special handling to be aligned with FogROS2-SGC. For example, gRPC requires the IP addresses of both robot and cloud for bi-directional message passing.

E. Compute and Memory-Efficient SGC router

FogROS2-SGC can be deployed on low-power robots under memory and compute constraints, so an efficient implementation of routing algorithm in Section IV-B is crucial to the overall performance of the system. Fig. 6 shows an architecture of SGC router. An idiomatic workflow of the router implementation is to (1) receive data from ROS2/network, (2) decide which network connection to forward, and (3) forward data to ROS2/network. Because FogROS2-SGC needs to be extensible to heterogeneous network protocols, the router needs to maintain many simultaneous network connections, ranging from ROS2's publish/subscribe protocol to general network protocol such as DTLS.

Because low-power robots run under memory constraints, memory copying operations and synchronization operations

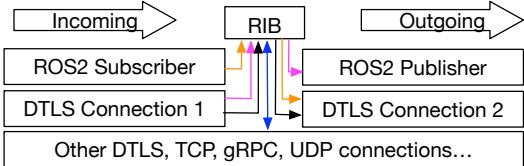


Fig. 6: SGC router architecture. (Orange) Subscribe to a local ROS2 network and publish to FogROS2-SGC routing network. (Magenta) Receive from FogROS2-SGC routing network and publish to local ROS2 network. (Black) Intermediate SGC router that facilitates message routing. SGC router asynchronously reads, writes, and manages all the network connections. All the message passing (arrows) is zero-copy and does not require movement of actual messages.

(such as mutex) are expensive. SGC router is implemented in Rust [41] to eliminate memory copying operations and the need for synchronization. Rust is a programming language that features a single ownership model: every data object has a single owner, and passing the data is moving the ownership from one variable to another. As a result, it prevents race conditions and reduces data copying by enforcing the passing of data objects by references instead of values.

Robots with few CPU cores usually have low network performance, because network operations are usually *blocking*, where the entire packet processing halts and waits for the network operations to finish. FogROS2-SGC improves CPU utilization by leveraging asynchronous Rust interfaces [42]. Asynchronous interfaces are non-blocking, removing the network operations out of the critical path of message processing.

V. EVALUATION

We evaluate FogROS2-SGC on system benchmarks to show how it performs over alternative designs and on robotics benchmarks to show how robotics applications benefit from FogROS2-SGC. We also demonstrate FogROS2-SGC on four physical robot arms running Fleet-DAGger [4], a multi-robot learning application. In Sections V-A and V-B, we use an Intel NUC with an Intel® Pentium® Silver J5005 CPU @ 1.50 GHz with a 5 Mbps network connection to act as the robot. The robot is connected with a Standard DS3 v2 cloud instance (4 vCPUs, 14 GiB memory) on Microsoft Azure. The robot is located at California (west coast of US), and the cloud server is located at Virginia (east coast of US).

A. System Benchmarks

We evaluate the performance of FogROS2-SGC’s message processing latency and throughput against other distributed ROS2 systems. Messages are sent in binary with type `sensor_msgs/CompressedImage` and response with string type `std_msgs/String`. We compare against the following baselines (1) **VPN**: We use Wireguard VPN [43], which is the same VPN as FogROS2 [19] (2) **Rosbridge**: Rosbridge is the most commonly used websocket proxy that allows non-ROS code to interact with ROS code. We use Rosbridge in combination with Rosduct in the same way as in FogROS [18]. (3) **Capsule**: We use Capsule, a software switch inspired by Netbricks [44], to emulate the design of FogROS2-SGC. We also implement rosduct [8] in ROS2 that converts between ROS2 and network traffic. The detailed

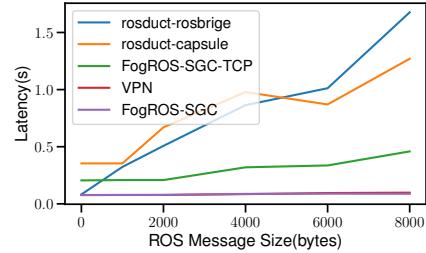


Fig. 7: **Message round trip latency to the cloud** (lower is better). Latency is averaged over more than 50 packet window. FogROS2-SGC is 19 times faster than rosbridge baseline for 8000 byte message.

Protocol	Throughput (msg/second)
Original ROS	330.43
SROS2	320.17
Rosduct-Rosbridge	152.79
FogROS2-SGC-TCP	268.03
FogROS2-SGC	320.40

TABLE I: **Message throughput evaluation of FogROS2-SGC** (higher is better). Every message is 1000 bytes. The throughput of FogROS2-SGC is near native performance while adding secure and global connectivity and 2.1 times higher than rosbridge.

description and implementation can be found in FogROS-G [45]. **FogROS2-SGC** uses the default DTLS network protocol. We include **FogROS2-SGC-TCP** that uses TCP instead of DTLS as a variant.

ROS2 Message Latency: We measure the Round Trip Time (RTT) between when a robot publishes a ROS2 message to the cloud and when data is received by the cloud, which echoes a short message on a separate ROS2 topic. The RTT also includes the time of parsing the messages and analyzing the latency. The result can be found in Fig. 7. FogROS2-SGC with DTLS has similar performance as VPN, which has 0.076s round trip latency for small messages. FogROS2-SGC is 10.2% faster than VPN for 8000 byte messages (0.088 vs 0.097). FogROS2-SGC is 19× faster than rosduct-rosbridge (0.088 vs 1.67). There are two reasons for this: (1) Rosduct is implemented in Python and provably slower than Rust. It uses blocking network operations while FogROS2-SGC uses non-blocking asynchronous network operations for sending and receiving data. (2) Rosbridge requires serialization of binary messages in JSON, which require more bytes and lead to larger messages.

ROS2 Message Throughput: Message throughput is measured by the number of messages processed per second. Different from other experiments, throughput is measured on the local area network connected with Ethernet, in order to prevent network bandwidth from being the bottleneck. Table I shows the message processing throughput. FogROS2-SGC achieves near-native throughput as ROS2 and incurs only 3% overhead due to the security and conversion to a unified message format. FogROS2-SGC has 2.1× higher throughput than rosbridge, because rosbridge requires more bytes to serialize binary strings.

Startup and Advertisement Time: In a RIB that has 10,000 routing records, the average time for publishing a name to the RIB takes 4ms and subscribing to a name from

Scenario	vSLAM	Grasp Planning	Motion Planning			
	fr/xyzl	fr/loop	raw matrix	Compressed	Apartment	Cubicle
rosduct-rosbridge	10.31	10.29	20.3	13.67	0.08	0.08
VPN	1.16	1.45	5.7	1.47	0.07	0.07
FogROS2-SGC-TCP	1.19	1.57	8.4	1.58	0.07	0.07
FogROS2-SGC	1.15	1.42	-	1.45	0.07	0.07

TABLE II: **Network latency of FogROS2-SGC on cloud robotics applications** (lower is better) FogROS2-SGC is better than rosduct-rosbridge and VPN on vSLAM and compressed grasp planning. We conducted motion planning on other scenarios (Home, TwistyCool) and the latency is the same.

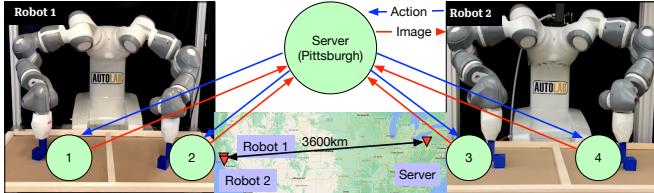


Fig. 8: **The experiment setup of Fleet-DAGger.** Two ABB YuMi robots located in two separate buildings in Berkeley utilize computation from a server located in Pittsburgh for an image based block pushing task.

RIB takes 2ms. The average startup time from starting a program to receiving the first message takes 2.4 ms.

B. Cloud Robotics Application Benchmarks

We evaluate the network latency of FogROS2-SGC with 3 example cloud robotics applications: SLAM with ORB-SLAM2 [46], Grasp Planning with Dex-Net [47], and Motion Planning with Motion Planning Templates (MPT) [48]. The detailed description of these benchmarks can be found in [18].

As detailed in Table II, although FogROS2-SGC can scale to multiple robots and provide fine grained access control for the robots, it demonstrates even better point-to-point performance than VPN in the vSLAM and grasp planning experiments. FogROS2-SGC is 9.42 times faster than rosbridge-rosduct on compressed grasp planning images. However, FogROS2-SGC cannot reliably transmit large and uncompressed grasp planning matrices. The raw matrix after serialization is larger than 13MB. We observe a significant amount of lost and out of order messages because the default transport protocol of FogROS2-SGC is DTLS over UDP and the communication channel does not recover from lost and out of order messages. Although transmitting such large message within single ROS2 message is rare, users can choose other supported transport protocols (such as TCP, gRPC) to meet the requirement of their applications.

C. Case Study: Fleet-DAGger

We apply FogROS2-SGC to the control of a fleet of 4 physical robot arms, an increasingly relevant setting in robotics and the third motivating example in Fig. 1. We use the physical experiment setup from Fleet-DAGger [4], where each robot simultaneously performs an image-based block-pushing task (see Fig. 1C). The task is to repeatedly push a cube to a goal region randomly generated in the image, where a new goal is sampled from the reachable workspace upon reaching the previous goal. The 4 workspaces have an identical setup (but different block positions and

Communication System	Server Location	Communication Time (s)
SSH + SFTP	Berkeley, CA Pittsburgh, PA	0.86 -
FogROS2-SGC	Berkeley, CA Pittsburgh, PA	0.31 0.58

TABLE III: **Communication time of SSH+SFTP and FogROS2-SGC** (lower is better). FogROS2-SGC with TCP reduces the communication time per experiment step (i.e., one simultaneous action on the 4 arms) by 64% when compared to SSH+SFTP, and has 33% lower communication time than SSH+SFTP in Berkeley even if the server is moved to Pittsburgh. SSH does not work if the server is in Pittsburgh due to a university firewall restriction.

goals) to enable the aggregation of each robot’s data into a shared dataset and training of a single shared policy on this dataset, as is typical in fleet learning [4]. When autonomous control is unreliable, the robots fall back on and learn from remote human teleoperation, where global connectivity can dramatically increase the number of available humans. The arms belong to two bimanual ABB YuMi robots in two different labs about 1km apart with separate local area networks. To test global connectivity, compute is off-loaded to a separate node in a third local area network at Carnegie Mellon University 3600 km away, where the robot nodes send images of the current state and receive actions to execute.

In a previous implementation, Hoque et al. [4] use Secure Shell (SSH) and Secure File Transfer Protocol (SFTP) to communicate between robots and the centralized compute node and Python multiprocessing to enable simultaneous execution. This approach requires storing all SSH credentials at a single node (a security concern), writing image data to the file system of all nodes at every timestep, complex asynchronous programming, and restricting all node locations to within the university campus firewall. To mitigate these issues, we (1) re-implement the communication system with ROS2 and (2) seamlessly connect all nodes with FogROS2-SGC with TCP by modifying only a single configuration text file on each node. Relative to the previous implementation, the FogROS2-SGC implementation reduces communication time by 64% (Table III), where communication time includes image transmission latency and synchronization across all arms but not machine learning or arm motion. FogROS2-SGC also reduces communication time by 33% relative to the initial implementation even when the robots are in Berkeley, CA and the server is moved to Pittsburgh, PA. Note that the SSH method does not work between Berkeley and Pittsburgh due to university network firewalls [49]. A diagram of the system architecture is in Fig. 8.

VI. CONCLUSIONS AND FUTURE WORK

We present FogROS2-SGC, an extension of FogROS2 that securely connects robotics components across different physical locations and networks. One limitation of FogROS2-SGC is that users are unable to use retransmission and Quality-of-Service (QoS) mechanisms provided by DDS for inter-ROS2 network communication. However, users can flexibly choose any supported transport protocol (e.g., TCP and gRPC). FogROS2-SGC will support other ROS2 communication patterns (e.g., services and actions) in the future.

ACKNOWLEDGEMENT

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab. The authors were supported in part by donations from Bosch, Toyota Research Institute, Google, Siemens, and Autodesk and by equipment grants from PhotoNeo, NVidia, and Intuitive Surgical. The research is also supported by C3.ai Digital Transformation Institute for AI resilience.

REFERENCES

- [1] 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.
- [2] J. Ichnowski, K. Chen, K. Dharmarajan, S. Adebola, M. Danielczuk, V. Mayoral-Vilches, H. Zhan, D. Xu, R. Ghassemi, J. Kubiatowicz, et al., “FogROS2: An adaptive and extensible platform for cloud and fog robotics using ros 2,” *arXiv preprint arXiv:2205.09778*, 2022.
- [3] “FogROS2 Official ROS2 Repository,” <https://index.ros.org/p/fogros2/>.
- [4] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg, “Fleet-DAGger: Interactive robot fleet learning with scalable human supervision,” *Conference on Robot Learning (CoRL)*, 2022.
- [5] S. Sumikura, M. Shibuya, and K. Sakurada, “OpenVSLAM: A versatile visual slam framework,” in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM ’19. New York, NY, USA: ACM, 2019, pp. 2292–2295. [Online]. Available: <http://doi.acm.org/10.1145/3343031.3350539>
- [6] A. S. Eddington, *The mathematical theory of relativity*. The University Press, 1923.
- [7] V. Mayoral-Vilches, R. White, G. Caiazza, and M. Arguedas, “SROS2: Usable cyber security tools for ros 2,” *arXiv e-prints*, pp. arXiv–2208, 2022.
- [8] “rosdust,” <https://github.com/uts-magic-lab/rosdust>.
- [9] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, “ROS and Rosbridge: Roboticists out of the loop,” in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012, pp. 493–494.
- [10] “Scalable Distributed Robot Fleet With Fast DDS Discovery Server,” <https://husarnet.com/blog/ros2-dds-discovery-server>, accessed: 2023-03-1.
- [11] N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, and K. Goldberg, “A cloud robot system using the dexterity network and Berkeley robotics and automation as a service (BRASS),” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 1615–1622.
- [12] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, “Cloud-based robot grasping with the google object recognition engine,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 4263–4270.
- [13] P. Li, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Goldberg, “Dex-Net as a service (DNaaS): A cloud-based robust robot grasp planning system,” in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, 2018, pp. 1420–1427.
- [14] B. Kehoe, D. Berenson, and K. Goldberg, “Estimating part tolerance bounds based on adaptive cloud-based grasp planning with slip,” in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, 2012, pp. 1106–1113.
- [15] B. Kehoe, D. Berenson, and G. Ken, “Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 576–583.
- [16] B. Kehoe, D. Warrier, S. Patil, and K. Goldberg, “Cloud-based grasp analysis and planning for toleranced parts using parallelized Monte Carlo sampling,” *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 455–470, 2014.
- [17] M.-L. Lam and K.-Y. Lam, “Path planning as a service PPaaS: Cloud-based robotic path planning,” in *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2014, pp. 1839–1844.
- [18] K. E. Chen, Y. Liang, N. Jha, J. Ichnowski, M. Danielczuk, J. Gonzalez, J. Kubiatowicz, and K. Goldberg, “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.
- [19] J. Ichnowski, W. Lee, V. Murta, S. Paradis, R. Alterovitz, J. E. Gonzalez, I. Stoica, and K. Goldberg, “Fog robotics algorithms for distributed motion planning using lambda serverless computing,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 4232–4238.
- [20] R. Anand, J. Ichnowski, C. Wu, J. M. Hellerstein, J. E. Gonzalez, and K. Goldberg, “Serverless multi-query motion planning for fog robotics,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE, 2021.
- [21] G. McGrath and P. R. Brenner, “Serverless computing: Design, implementation, and performance,” in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2017, pp. 405–410.
- [22] G. Swamy, S. Reddy, S. Levine, and A. D. Dragan, “Scaled autonomy: Enabling human operators to control robot fleets,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5942–5948.
- [23] A. K. Tanwani, R. Anand, J. E. Gonzalez, and K. Goldberg, “RI-LaaS: Robot inference and learning as a service,” *IEEE Robotics & Automation Letters*, vol. 5, no. 3, pp. 4423–4430, 2020.
- [24] S. Bauer, F. Widmaier, M. Wüthrich, N. Funk, J. U. D. Jesus, J. Peters, J. Watson, C. Chen, K. Srinivasan, J. Zhang, J. Zhang, M. R. Walter, R. Madan, C. B. Schaff, T. Maeda, T. Yoneda, D. Yarats, A. Allshire, E. K. Gordon, T. Bhattacharjee, S. S. Srinivasa, A. Garg, A. Buchholz, S. Stark, T. Steinbrenner, J. Akpo, S. Joshi, V. Agrawal, and B. Schölkopf, “A robot cluster for reproducible research in dexterous manipulation,” *arXiv preprint arXiv:2109.10957*, 2021.
- [25] J. Z. Lim and D. W.-K. Ng, “Cloud based implementation of ROS through VPN,” in *Int. Conf. on Smart Computing & Communications (ICSCC)*. IEEE, 2019, pp. 1–5.
- [26] S. S. H. Hajjaj and K. S. M. Sahari, “Establishing remote networks for ROS applications via port forwarding: A detailed tutorial,” *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417703355, 2017.
- [27] eProsima, “Fast DDS,” <https://www.eprosima.com/index.php/products-all/eprosima-fast-dds>.
- [28] “RTI Connext DDS,” <https://www.rti.com/products>.
- [29] “Integrating ROS2 with Eclipse zenoh,” <https://zenoh.io/blog/2021-04-28-ros2-integration/>, accessed: 2021-02-15.
- [30] A. B. M. Pereira, R. E. Julio, and G. S. Bastos, “ROSRMote: Using ROS on cloud to access robots remotely,” in *Robot Operating System (ROS)*. Springer, 2019, pp. 569–605.
- [31] B. Xu and J. Bian, “A cloud robotic application platform design based on the microservices architecture,” in *Int. Conf. on Control, Robotics and Intelligent System*, 2020, pp. 13–18.
- [32] E. Boasson, A. Corsaro, and H. van t Hag, “Eclipse Cyclone DDS,” <https://projects.eclipse.org/projects/iot.cyclonedds>.
- [33] “Google Remote Procedure Call,” <https://grpc.io/>.
- [34] M. Myers, C. Adams, D. Solo, and D. Kemp, “Internet x. 509 certificate request message format,” Tech. Rep., 1999.
- [35] “Signature and hash algorithms for TLS and DTLS,” <https://www.ibm.com/docs/en/zos/2.5.0?topic=support-signature-hash-algorithms>.
- [36] A. W. Appel, “Verification of a cryptographic primitive: Sha-256,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 37, no. 2, pp. 1–31, 2015.
- [37] N. Mor, R. Pratt, E. Allman, K. Lutz, and J. Kubiatowicz, “Global data plane: A federated vision for secure data in edge computing,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1652–1663.
- [38] “Multicast DNS RFC 6762,” <https://www.rfc-editor.org/rfc/rfc6762.html>.
- [39] J. Daemen and V. Rijmen, “AES proposal: Rijndael,” 1999.
- [40] “RSASSA-PSS RFC 4056,” <https://www.rfc-editor.org/rfc/rfc4056>.
- [41] N. D. Matsakis and F. S. Klock II, “The rust language,” in *ACM SIGAda Ada Letters*, vol. 34, no. 3. ACM, 2014, pp. 103–104.
- [42] “Tokio,” <https://tokio.rs/>.
- [43] “wireguard VPN,” <https://www.wireguard.com/>.
- [44] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, “{NetBricks}: Taking the v out of {NFV},” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 203–216.
- [45] 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.

- [46] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras,” *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [47] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, “Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” in *Proc. Robotics: Science and Systems (RSS)*, 2017.
- [48] J. Ichnowski and R. Alterovitz, “Motion planning templates: A motion planning framework for robots with low-power CPUs,” in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.
- [49] “UC Berkeley Minimum Security Standard,” <https://security.berkeley.edu/policy/minimum-security-standards-networked-devices-mssnd>.