5-2024

Multi-Domain Secure DDS Networks for Aerial and Ground Vehicle Communications

Daniel Pendleton
dpendle@clemson.edu

Multi-Domain Secure ROS2/DDS Networks For Aerial And Ground Vehicle Communications.

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Electrical Engineering
Master of Science

---

by
Daniel Scott Pendleton
May 2024

Accepted by:
Dr. Kuang-Ching Wang, Committee Chair
Dr. Linke Guo
Dr. Harlan Russell

Abstract

According to the U.S. Department of Defense, it's estimated that there are over 11,000 units of Unmanned Aerial Vehicles/Systems (UAVs/UASs) deployed for both domestic and international operations[10]. To keep up with modern warfare and emerging threats, the next generation of unmanned systems must address these drawbacks with improvements in networked communications and security. UAV networks must be secure and be able to detect and remove malicious agents in the event of an infiltration. They must also be able to produce large amount of data without creating bottlenecks or slowdowns in order to maintain real-time capabilities.

The military and other industries have shown increased interest in implementing Robot Operating System 2 (ROS 2) as the application-level control for their autonomous UAVs. The network layer is controlled by the Data Distribution System (DDS) which manages the messaging system for each robot. There are still questions regarding DDS' ability to meet the needs of military applications, specifically with respect to the system's security features and potential for scalability. For military expectations, DDS must be able to transmit secure data across multiple Wide Area Networks (WANs), a task of past DDS's original implementation.

This thesis, we will analyze the security and scalability capabilities of DDS through the application of a ROS 2 ecosystem. This work explores the network performance impacts of multiple DDS-facilitated network topologies. These topologies include Local Area Networks (LANs) and WANs connected together via multi-hop routing. We investigate the effects of scaling network size and increasing the payload size of the transmitted data. These experiments will also reveal the impacts of varying topologies on ROS 2 performance. This work contributes an evaluation testbed for DDS networks to show its limitations and suggests ways to create accurate mission scenarios for military devices.

Dedication

First and foremost, I want to dedicate this thesis to my Mom and Dad. They knew my dream was always to come to Clemson since I was young and I've always wanted to make them proud even if I have to work twice as hard to get there. I also would love to dedicate this to my fiancée and soon-to-be wife Caitlyn. She has been with me through thick and thin on my academic journey going back to high school together, to our first year of college together, and then doing long distance for three more years until now. She's my best friend and I'm happy to dedicate this thesis to her. I also want to share love with my sister Kate and my brother Buck, I always try to make you proud in anything I do and appreciate the constant love you guys give me. Thank you for the opportunity.

Acknowledgments

First, I cannot thank my advisor, Dr. KC Wang, enough for the guidance and leadership needed in this incredible academic journey. He has done so much to help my development as a graduate student and network researcher. I appreciate all the extra time he's given to help me think out these challenging problems and push me to grow, not only as an engineer but as a professional in all facets of my life. I'm extremely grateful for the opportunity to work closely with him on the VIPR-GS project.

I would also like to give many thanks to Dr. Guo and Dr. Russell as well. These two were pivotal in getting me to come to Clemson for graduate school, and I would not have been here without them. I greatly appreciate the opportunity Dr. Guo gave me as an undergraduate to begin my research career, which propelled me to this point. I would also like to thank my entire committee for giving their time and wisdom on my research. The entire ECE department deserves recognition because none of this would be possible without them.

Also, I would like to thank all my fellow students and researchers throughout this process. Without the guidance and help of people like Brandon Rice, Ben Formby, Acheme Acheme, and Charles Kowalski this project would not have been possible in the time it was done. I will forever be grateful for their help and hope the best for their futures in whatever they pursue.

Finally, I would like to thank my family and friends for supporting me from the very beginning. Especially my fiance Caitlyn and my entire family, who have given me the drive to make them as proud as I possibly can with their unconditional love. I finally want to say thank you, God, for giving me this opportunity in life. Nothing would be possible without Him opening doors like this in my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The development of Unmanned Systems has been greatly shaped by technological progress and strategic requirements. The U.S. Department of Defenses Unmanned System Integrated Roadmap (2017–2042) emphasizes the importance of interoperability and network security when deploying these systems in domains like air, surface and subsurface [10]. Thanks to advancements in miniaturization, autonomous, and networking technologies, these systems have expanded their roles in these domains while boosting capabilities and adapting to the challenges of modern warfare.

Mini UAVs represent an advancement due to their cost-effectiveness, ease of deployment, and ability to access areas that larger systems cannot reach. However, their smaller size does come with limitations, such as reduced payload capacity and shorter communication ranges. Overcoming these obstacles has been made possible through the use of swarm tactics, enabling UAVs to collaborate towards a shared goal and increasing their effectiveness in missions requiring extensive reconnaissance and search operations.

Integrating UAVs into activities has presented its own set of obstacles. A noteworthy incident during a conflict that involved a breach in security where Ukrainian intelligence successfully infiltrated the controls of a UAV [5]. This breach not only exposed information but also led to direct targeting and destruction of a Russian base through artillery strikes. In the realm of aerial vehicle operations, it is imperative to address vulnerabilities in communication links and data security as they pose significant risks to mission integrity and personnel safety.

As the use of drones continues to grow in civilian settings, ensuring communication channels and preventing unauthorized access are top priorities. Traditional systems like the Robot Operating

System (ROS) have historically lacked security measures such as encrypted communications. Our study presents an approach to enhancing security in ROS 2 nodes by implementing a Public Key Infrastructure (PKI) in DDS Secure. This strategy helps encrypt data transmissions to nodes, bolstering the security framework needed for robotic system operations in environments where robust security features are essential.

The reliability of network communications is crucial for vehicles' effectiveness in modern warfare reconnaissance and defense strategies. This thesis examines ROS 2 and DDS network capabilities within a Wide Area Network (WAN) setup. It focuses on understanding how hops between sites impact network performance under security measures. By conducting tests to send sizes of data packets through network setups, our goal is to identify potential bottlenecks and evaluate how well the real-time systems can handle slowdowns. More interest from the industrial sector in using ROS 2 for managing vehicle fleets, along with DDS's role in managing the messaging system, highlights the need for a network that not only focuses on security but also demonstrates efficient scalability beyond what DDS was initially designed for.

This study focuses on examining the security features of DDS middleware within a ROS 2 environment, especially looking at how it scales and performs in scenarios involving multi-hop routing across local area networks (LANs) and wide area networks (WANs). We will also explore how different data packet sizes affect the network and how removing compromised devices dynamically impacts its operation. Our goal is to provide a testing platform for DDS networks, pinpointing their constraints and proposing ways to create mission scenarios for military use. This research emphasizes the role of network management and security measures in ensuring that future unmanned systems operate effectively and securely.

1.1 Robot Operating System (ROS)

The Robot Operating System (ROS) is an modular framework created by Robotics, previously known as the Open Source Robotics Foundation [28]. It aims to speed up the development of applications and has gained popularity in the robotics community due to its flexibility in supporting software and libraries for a wide range of projects. The initial version, ROS 1, introduced a network architecture with a central master node that handles tasks like managing application registration, execution, storage of parameters, and logging message traffic [31]. This setup allows for the deploy-

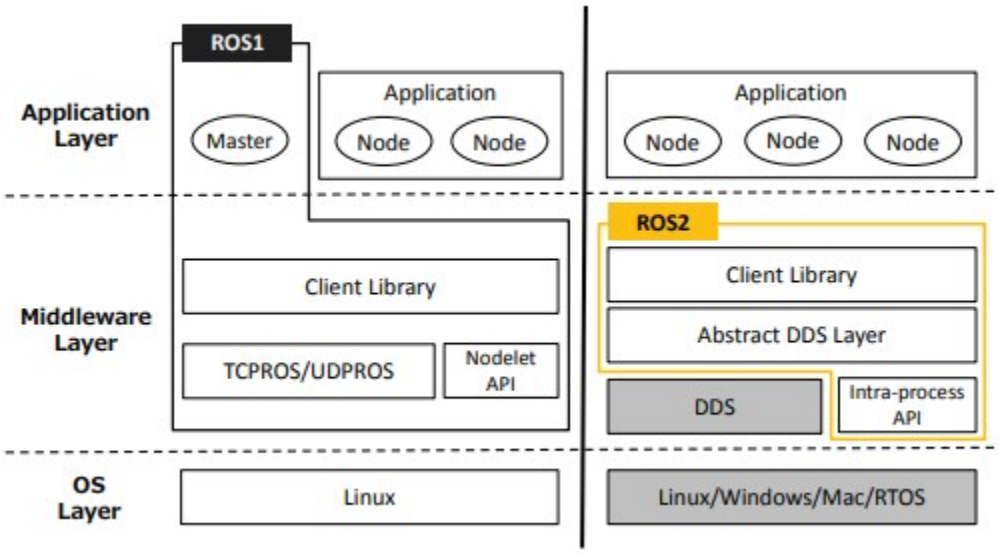

Figure 1.1: ROS1 and ROS2 architechture [25]

ment of software nodes tailored for functions such as executing commands, processing data, and managing sensor information [39]. These nodes communicate with each other through an XML RPC based API for activities like publishing topics, subscribing to them, and updating parameters [39]. Despite its use and the development of versions like ROS I for industrial use and ROS M for military applications to meet specific needs in different sectors, ROS 1 had certain limitations.

Its initial design, which was tailored for use on machines or small groups, faced challenges when it came to scaling up for distributed systems. The centralized network setup resulted in performance issues and made the system vulnerable to failure at points. Additionally, the lack of built-in security and the less-than-optimal messaging system for real-time operations severely restricted its usefulness in critical scenarios [32] [33].

To address these shortcomings, ROS 2 introduces an approach by utilizing the Object Management Groups (OMG) Data Distribution Service (DDS) to create a real time and secure communication framework. DDS is a recognized standard for real-time systems to missions and integrates the eProsima Fast RTPS protocol to enable efficient communication between publishers and subscribers even over unreliable channels like UDP [17]. This transition not only boosts scalability and reliability but also brings significant security enhancements, expanding ROS 2s potential applications across various sectors, including government and industry. With these advancements in

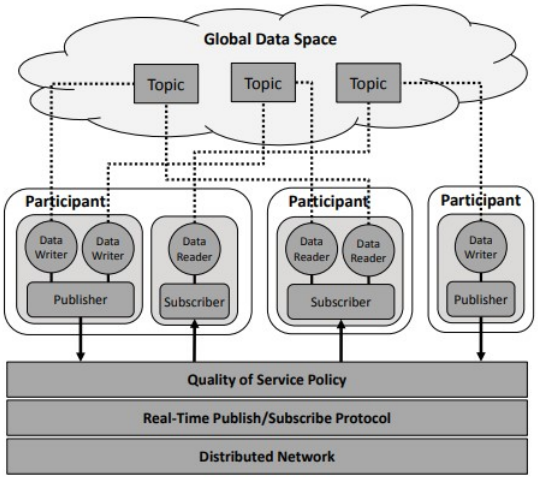

Figure 1.2: ROS2 Architechture for Data-centric publisher and subscribers [25]

place, ROS 2 successfully surpasses the limitations of its predecessor establishing a standard for the development of systems. The focus on security, the ability to process data in time, and scalability highlight how ROS has developed into a tool in the field of robotics. It is well-equipped to handle the requirements of security-conscious applications.

1.2 ROS2 and Data Distributive Systems

ROS 2's backbone for its communication for decentralized and distributed systems is DDS, which allows for a rich set of Quality of Services (QoS) policy that enables it to cater to specific needs for each robot fleet that specifies throughput, latency, and best-effort reliability. DDS utilizes the Data-centric Publish-Subcribe (DCPS) model to create a "global databus" that can be reached by any independent entity. This allows for efficient data distribution between each node that is referred to as a participant in DDS.

1.2.1 DDS layer

DomainParticipant: serves as a gateway and organizational entity in DDS, enabling applications to interact within a designated domain. Domains are established to separate and enhance communication, among application groups guaranteeing both isolation and streamlined data sharing. **Publisher:** serves the distributing data mechanism for transmitting information. It

manages DataWriters, each tasked with sharing data on specific Topics, facilitating the spread of information to interested individuals.

Subscriber: Opposite of the Publisher, it's designed to receive data from the network. It represents one or more DataReaders, allowing a DomainParticipantto collect and handle various types of data.

DataWriter: Essential for publishing, a DataWriter is an object that is always used by the DomainParticipant to write data via the publisher. The DataWriter publishes predefined data types.

DataReader: Connects a Subscriber as an object and plays a crucial role in receiving and accessing published data. Data types coming from the DataWriter must match the data type expected at the DataReader for accuracy and compatibility.

Topic: Describes the classes used to define communication topics and data types between the DataWriter and DataReader. Each Topic is uniquely identified by a name and is associated with a specific data type, establishing a clear and structured data communication channel.

1.2.2 Public key infrastructure (PKI)

In 1970, Feistel [18] designed the first automated, computerbased cypher system for securing the digital connection between two computers (called a data bank) for IBM Research. As this was just a concept, the following year, Smith [37], another IBM Research employee, implemented a cryptographicsystem named Lucifer, heavily influenced by Feistel's work. Lucifer introduced a single 128-bit cipher key that enciphers (encrypts) and deciphers (decrypts) messages in blocks of 16 bytes. Using Smith's system, for the first time, two computers connected together could have data sent between them without the fear of bad actors viewing the plaintext data stream. Even worse, the compromised data stream could be recorded as a copy on tape for later illicit use. These works introduced a technology still used today: symmetric keys, mechanisms that both encrypt and decrypt data using the same transformation. Symmetric keys, such as the Data Encryption Standard (DES) [23] [27] (a successor to Lucifer that also uses a Feistel network [38]) and its successor, the Advanced Encryption Standard (AES) [9], are used to provide confidentiality and integrity of data [7] [22].

Although symmetric keys are computationally efficient, they are severely limited in providing authenticity between two computers. One challenge is the single key-value must be distributed (and

is hence susceptible to being compromised) to the computer receiving the message [22]. To combat this issue and develop a method for providing authentication, Diffie and Hellman [11] developed a public key distribution system, later to be recognized as the Diffie-Hellman key exchange. Their concept was to have each computer use a keypair, consisting of a public key and a private (or secret) key, rather than to share a single cipher key. Computers can use these keys in two notable ways, each independently insignificant for providing broad security. However, when these methods are combined in a single system, they provide a vast amount of security. This system, called a Public Key Infrastructure, is the basis of modern cybersecurity tools such as SSL/TLS, which is used to secure users' connections to over 88.5 million websites [3].

In the following scenarios, we describe two networked computers named Sender and Receiver. The public keys of all computers are public, and thus can be accessed or distributed to each other.

1) Encryption Using a Public Key: The first method in which keypairs can be used is to have Sender encrypt a plaintext message using Receiver's public key. The message is now ciphertext and Receiver's private key is the only key that can decrypt the message. Assuming Receiver's private key is not compromised, the message can be sent while providing confidentiality. Receiver decrypts the message using Receiver's private key. This is similar to how symmetric encryption works, although the issue of distributing the symmetric key is avoided. Like symmetric encryption, this method fails to provide authenticity and integrity of the message.

2) Signing Using a Private Key: Unlike the first method, Sender now signs (encrypts) a plaintext message using its own private key. The message is now ciphertext and Sender's public key is the only key that can decrypt the message. Since all computers have access to Sender's public key, the message can be received and decrypted (and thus verified that Sender sent the message), providing authenticity and integrity to the system. The message could not have been created or modified by anyone other than Sender assuming Sender's private key has not been compromised. These two methods, and additional features, are combined to create a Public Key Infrastructure (PKI). A PKI additionally uses certificates, ideated by Kohnfelder [24], to digitally sign each public key, providing a mechanism for a computer to establish trust with other computers through a mutually agreed centralized entity called a Certification Authority (CA) [20]. Standards such as X.500 [1], and later X.509 [2] (used in this paper), were created by the International Telecommunication Union (ITU) to define the creation, structure, and functionality of certificates. In addition to providing trust to the system, a PKI's CA can revoke a computer's trust by adding the computer's certificate to a

Certificate Revocation List (CRL) as proposed originally in 1994 by Berkovits et al [8]. CRLs are public, that is either accessible or distributed to each computer in the network, so other computers can be notified not to trust the computer with the revoked certificate.

In the following example, we add a computer to act as the CA, to our previous example scenario. This CA has the ability to create and distribute a CRL to Receiver.

3) An Example Use of PKI: When entering the network, both Sender and Receiver share their public keys with the CA. The CA creates a X.509 certificate for each public key and signs both certificates with its (the CA's) private key. Thus, each public key contained in the certificate cannot be modified unless the CA's private key has been compromised. Each certificate is transmitted back to the computer with ownership of the respective public key. To establish trust and authentication of Sender, Sender sends both its certificate and its public key to Receiver. If Receiver determines the certificate to match with a revoked certificate found in the CRL, it will prevent the request to establish trust from succeeding. Otherwise, this process will continue. Using the CA's public key, Receiver then decrypts the public key contained in the certificate and compares it with the public key changed by Sender. If both keys are identical, Receiver can confirm the messages transmitted from Sender have both authenticity and data integrity. This process is repeated, now having Receiver send Sender its (Receiver's) certificate and public key. If both Sender and Receiver succeed in establishing trust, Sender can then create a symmetric key and encrypt it with Receiver's public key. If Receiver's public key has not been compromised, the only computer that can decipher the symmetric key sent over the network is Receiver. Upon receiving the encrypted symmetric key, Receiver decrypts the symmetric key and is ready to receive incoming messages encrypted with the symmetric key. Sender can then send these encrypted messages over the network securely, providing confidentiality

1.3 ROS 2 Security

Recognizing the importance of security, in the changing world of robotics particularly as systems become more integrated into critical environments requires a thorough exploration of the improvements brought about by ROS2. Although ROS1 set a foundation it failed to address the intricate cybersecurity needs of advanced applications leaving unmanned systems vulnerable to different security risks. This realization led to the development of ROS2, which not prioritizes network

security through measures such as identity verification and resource access control but also seamlessly incorporates the functionalities of Secure ROS (SROS) software package directly into its core system, a feature that was previously an addition in ROS1.

The introduction of ROS2 marks a shift towards a robotic environment where the collaboration between Registration Authority (RA) and Certificate Authority (CA) serves as the cornerstone of its security architecture. The RA has the ability to automate or supervise identity verification through methods ranging from email based registration to more stringent in person verification processes similar to passport issuance. Similarly the CA plays a role, in digitally signing certificates like a would ensuring their legitimacy and integrity. These certificates are stored in a directory, for access to a phone directory enhancing the systems security by providing verifiable credentials.

Key to the effectiveness of the DDS security extension in ROS2 are two core policies; the Domain Governance Policy and the Participant Policy. The Domain Governance Policy utilizes XML file structures to manage domain and topic elements using values to activate or deactivate security features. Once signed digitally by a CA, this policy serves as a cornerstone for guaranteeing authorized access and securing communication channels. Participant Policy outlines permissions for domain participation and access controls requiring signatures from a CA for implementation. This unique dual CA structure in the model introduces a degree of flexibility in identity verification and policy enforcement ensuring a secure and resilient network.

Furthermore plugins are integrated into the DDS framework, for authentication, access control, cryptography, logging and data tagging. These plugins support messaging and thorough activity monitoring to enhance ROS2s security landscape further. In this environment both keystore management systems and enclaves play roles in establishing a robust security framework. The keystore is designed with private directories to effectively manage security credentials.

Authentication: Allows for the identity of the Publishers/Subscribers on the network participating.

Access Control: Allows which topics are able to access the published topic or who can subscribe to certain topics as well.

Cryptographic: Implements of cryptographic operations including encryption, decryption, hashing, digital signatures, etc.

Data Tagging: Offers the ability to give extra tagging details to data samples issued by the data writer.

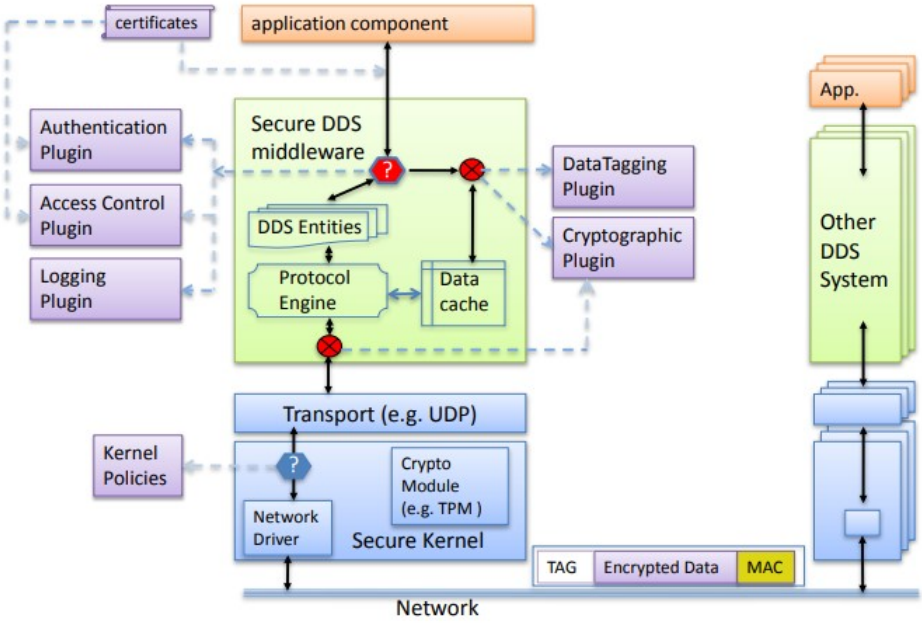

Figure 1.3: DDS security model from [26].

Logging: Enables the auditing of all security-related events within DDS.

The public folder acts as a place where certificates, from Certificate Authorities such as identity CA and permissions CA are stored in a read only manner for all ROS nodes to access. This setup ensures the widespread sharing of identity verification information. On the hand the private folder contains materials, like the CAs private keys, which must be deleted before installing the keystore on a target device to prevent unauthorized access risks.

The enclaves directory in SROS2 is significant as it holds security details for security enclaves—defined as processes or groups of processes that share the same identity and permissions. This approach of compartmentalization enables control over security credentials and permissions within the ROS2 system. When initializing the keystore security keys and governance documents are created for these enclaves, including files like identity certificate permissions for verifying enclave identities along with permissions certificate, governance documents (governance.p7s) and permission documents (permissions.p7s). Together these files. Enforce security policies and access controls for each enclave strengthening the security framework.

Crucial elements like certificates for identifying participants, RSA or ECDSA algorithms for encryption and Diffie Hellman Key Agreement, for symmetric key exchange all play essential

roles in this context. The access controls regulate what participants can do while the cryptographic functions, like encryption, decryption and key generation play a role in maintaining data security. Logging mechanisms keep track of security events with timestamps and data tagging to add layers of protection to data samples. This detailed setup, which will be visually explained in the figure supports exploration and secure communication between publishers and subscribers in the DDS network ensuring both data integrity and participant verification. With these enhancements ROS2 not addresses the security issues of its predecessor. Also establishes a higher cybersecurity standard for robotics leading the path towards safer and more dependable unmanned system deployments, in various fields.

1.4 RTPS secure submessage

Securing messages in DDS involves adding information to inform the recipient about the security measures (Transformation Kind) and the key identifiers (Sender Key ID) used, enclosed in a Crypto Header. Subsequently the Crypto Content submessage component serves as a cover for Serialized Payloads, RTPS submessages or entire RTPS messages depending on the level of security implemented. The applied transformation produces the Crypto Content, which is then supplemented by a Crypto Footer containing the Common MAC for ensuring message integrity. In cases where Origin Authentication Protection is enabled Receiver Specific MACs are also integrated into the Crypto Footer. Message security in DDS varies depending on how protection's applied affecting the secure delivery of messages. Here are detailed adjustments, within the RTPS protocol to facilitate entity communication and their associated implications.

1.4.1 Architecture of RTPS for Secure Traffic

The essence of the RTPS protocol lies in an RTPS message, which consists of a header and various submessages each with its unique header and elements. These submessages within a RTPS message can encompass a wide range of content including user data, heartbeat signals, acknowledgments and more. Security measures can be implemented at levels to protect specific message components.When the Governance Rule for data_protection_kind is set to a level than NONE Serialized Data Protection comes into play focusing on safeguarding the DataWriter. In this scenario the DataWriter ensures the security of the sample payload after serialization by storing it in a

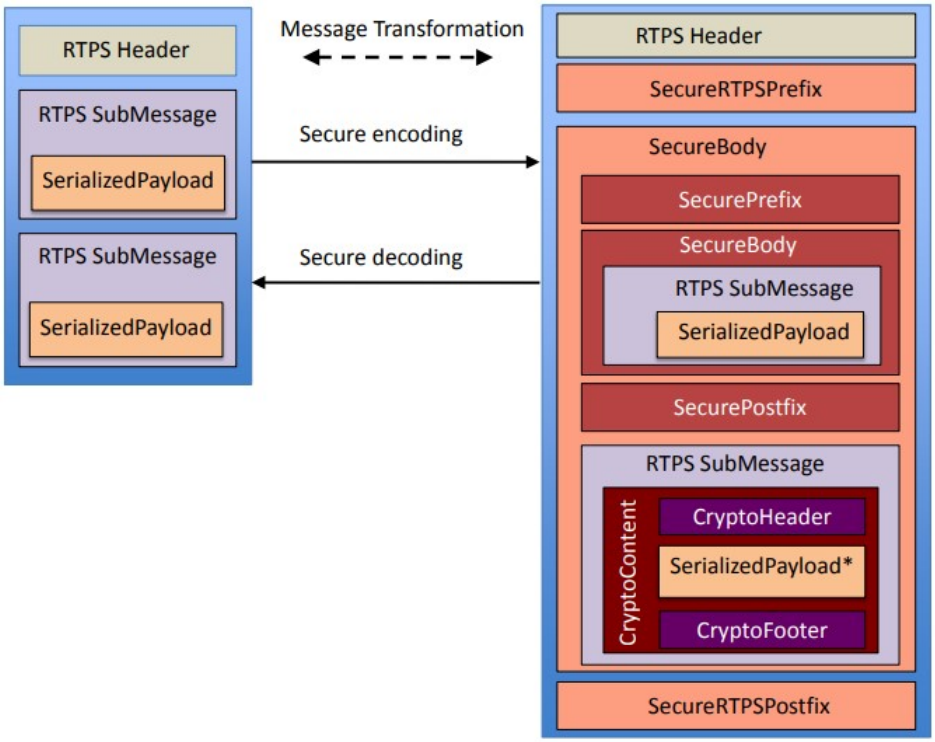

Figure 1.4: DDS security model from [26].

queue until it's ready for dispatch. This protection covers the payload within DATA submessages. Transforms the Serialized Payload into a secure format that comprises a Crypto Header, Serialized Payload/Crypto Content and Crypto Footer. Activating Submessage Protection by configuring the `metadata_protection_` kind Governance Rule affects messages from both DataWriters and DataReaders. Certain RTPS submessages undergo security measures before transmission through elements like Secure Prefix Submessage `(SEC_PREFIX)` Secure Body Submessage `(SEC_BODY)` with Crypto Content and Secure Postfix Submessage `(SEC_POSTEFIX)`, with Crypto Footer. RTPS Protection comes into effect by setting the rtps_protection_kind Governance Rule to secure not individual components but also the entire RTPS message before it is transmitted over the network. This thorough security protocol maintains the RTPS header encloses the complete message and the secure packet that is generated consists of an SRTPS Prefix, with the Crypto Header, a Crypto Content filled SRTPS Body and a Crypto Footer containing SRTPS Postfix.

1.4.2 Cryptographic Labeling in RTPS Messaging

When the sender prepares a message they add components like the Crypto Header and Crypto Footer which include MACs and important details for the recipient to generate the Session Key. The Crypto Header informs the recipient about the encryption method used and the specific keys utilized for message security. It consists of `transformation_id`, which combines Transformation Type with Sender Key ID, `session_id` for encryption operations and `initialization_vector_suffix` for AES-GCM (encryption) and AES-GMAC(authentication) methods. The Crypto Content contains data for messages that require both integrity and confidentiality protection (ENCRYPT) while the Serialized Payload carries plain text data for integrity protection only (SIGN). Completing the message structure the Crypto Footer includes the Common MAC and Receiver Specific MACs where necessary. The Common MAC ensures message integrity while Receiver Specific MACs offer origin authentication, crucial for communication, in multicast settings.

1.5 DDS Routing via ROS 2

DDS protocol is known for its ability to support UDP/IP multicast facilitating interaction, between diverse vendor source codes and enabling both wireless interoperability. This feature creates an ecosystem where applications can communicate efficiently regardless of their hardware or software origins. However when extending communication beyond a Local Area Network (LAN) to Wide Area Network (WAN) environments DDS faces challenges due to limitations imposed by Internet Service Providers (ISPs) in WANs. These limitations often restrict UDP flows and multicast traffic to prevent network flooding and maintain stability. Additionally the prevalent use of TCP for network traffic adds complexity by requiring TCP connections, between publishers and subscribers. This could potentially impact the efficiency of DDS core discovery mechanisms— the Participant Discovery Protocol (PDP) and Endpoint Discovery Protocol (EDP).These guidelines play a role, in how participants interact within the DDS network. PDP helps detect participants while EDP allows them to find each others topics of interest. When a publisher wants to share information across networks having to create TCP connections with each subscriber on a peer to peer basis can hinder the efficiency and scalability of the DDS framework.

To overcome these obstacles, recent studies and advancements from DDS vendors have led to the development of DDS routers. These routers act as bridges that enable communication between

network domains without requiring direct TCP connections. By routing DDS traffic these routers maintain the protocols efficiency. Expand its usability beyond single LAN setups. One notable solution among these innovations is eProsimas DDS Router, which offers an architecture for handling network DDS communication complexities. This router not overcomes limitations imposed by ISP regulations and TCPs prevalence outside LANs but also boosts the scalability and adaptability of DDS applications, across networking environments.

When it comes to DDS routing, both the Participant Discovery Phase and Endpoint Discovery Phase are particularly significant. During the Participant Discovery Phase, DomainParticipant sends periodic announcement messages that contain metadata, such as IP and port, about the participant. This phase plays a role in building a network of participants that can communicate with each other. In the Endpoint Discovery Phase, each participant learns about the data sharing and subscription details of other nodes so they gain insights into data topics and network structure [16]. These discovery phases are crucial in DDS's communication model allowing participants to efficiently locate and exchange information without setup requirements.

While DDS routers have advanced in connecting network domains and facilitating communication across environments there are still hurdles to overcome. One key issue revolves around the quality and dependability of documentation provided by vendors. Although routers like those from eProsima have enhanced DDS functionality in LANs and WANs their accompanying documentation may lack the depth required for understanding and successful implementation. This deficiency in guidance could hinder system architects and developers in harnessing the capabilities of DDS routers potentially impacting the deployment of DDS based systems in intricate network setups.

Moreover, ensuring seamless DDS communication across multiple network hops remains a challenge. The utilization of DDS routers to facilitate data transmission across diverse network segments each with its distinct characteristics and potential failure points presents a significant obstacle. The complexities involved in traversing hops become particularly pronounced when dealing with fluctuating network topologies and conditions underscoring the importance of routing solutions and clear implementation guidelines. The issues pertaining to documentation quality and the varying reliability of DDS routers in hop scenarios underscore the necessity for research efforts and community engagement aimed at enhancing these tools to optimize their effectiveness, within modern networking landscapes.

1.6 Contributions

In this thesis, we take a dive into examining the strengths and limitations of the DDS and ROS 2 in secure environments specifically looking at how well they handle transmitting messages across multiple devices from edge to edge in varying scenarios. Our goal is to assess how effective these technologies are, in real time situations that demand timely data delivery.

To accomplish this we establish a testing framework that replicates real world conditions enabling us to assess the performance, scalability and resilience of DDS and ROS 2 networks under various security constraints and network setups. This framework not only showcases the capabilities of these systems but also identifies potential weak points that could hinder their use in critical settings.

Additionally, we delve into the nature of security management in these distributed systems. By introducing a mechanism for adjusting security permissions in time we showcase the ability to swiftly isolate and remove compromised nodes from the network. This feature is essential for safeguarding the confidentiality and integrity of exchanged data, within the network in environments where security threats are constantly evolving. Also, our study adds to the development of guidelines and best practices, for setting up and managing DDS and ROS 2 systems in hop environments. These suggestions are aimed at helping system designers and developers optimize their setups for security, performance and resilience.

Moreover, our research findings have implications that go beyond DDS and ROS 2 technologies. By tackling the challenges of communication in hop networks we offer valuable insights into the wider realm of secure distributed systems. This work opens doors for research efforts and technological advancements that could strengthen the reliability and security of infrastructure and secure robotic ecosystems.

To sum up this thesis is not just highlights the capabilities of DDS and ROS 2 in secure multi-hop scenarios but also lays the groundwork for continuous enhancements and innovations in secure real-time communication, for essential applications.

Chapter 2

Related Work

2.1 Background

Recent research has focused on studying the impact of DDS security protocols, on communication efficiency in ROS 2 environments. Previous studies have looked into the latency and throughput effects of these protocols in both wire and wireless setups aiming to find a balance between performance and security. This research has provided insights for ROS 2 developers within LAN setups. At the time innovative work has introduced new router designs for DDS that help distribute data across Wide Area Networks (WAN). This advancement highlights the benefits of using DDS routers for network communication but does not fully address the complexities of security overhead and transmitting encrypted data across different networks.

A detailed examination outlined in [12] appraised DDS alongside ROS 2 contrasting data transfer with data that was safeguarded using encryption techniques like Rivest Shamir Adleman (RSA) 2048 bit and Elliptic Curve Cryptography (ECC) 256 bit. The results demonstrated a rise in delay and packet transmission overhead averaging 137 percent and 132 percent respectively when security measures were activated. This emphasizes the influence of security protocols on network performance a point also raised in observations from the 2018 ROSCon, where researchers underscored the latency increase linked to the integration of security features by DDS provider Real Time Innovations (RTI) [29].

Expanding on the research in [36] which examined ROS 2s ability to secure communications between UAVs and a ground control station (GCS) without concentrating on network performance

our study broadens this investigation to include the DDS security structure and the various service delivery configurations accessible, through ROS 2. While earlier studies often separated QoS from security considerations our research strives to combine these aspects. We aim to explore how the combination of QoS and security protocols affects the performance of ROS 2 networks delving into areas that have not been extensively covered in existing research. The next following papers are the ones that we felt had the most valuable information from so it would be best to explain in detail what they considered important in the security aspect and the DDS Routing Architecture.

2.2 Performance of Robot Operating System 2 with QoS and Cyber Security Configurations

The paper referred in [19] goes into investigating how well the ROS 2 performs, especially when combined with the DDS as its middleware. The main focus is, on how Quality of Service (QoS) settings in the middleware interface impact meeting strict delivery requirements across a dynamic network of unmanned systems. By conducting experiments under scenarios with different QoS configurations the study aims to understand how latency and throughput affect data transmission while also considering the effects of activating DDS security measures. This comparative evaluation against performance benchmarks aims to reveal the balance, between security measures and operational efficiency within ROS 2 networks.

The tests carried out on a ROS 2 network, which includes both a publisher and a subscriber offer insights, into how various QoS profiles and security configurations impact network efficiency. This study is groundbreaking in its examination of ROS 2s effectiveness in terms of service provision and security aspects. The key highlights of the study involve:

Analysis and experimentation: ROS 2 performance of various QoS profile combination for plaintext data traffic, focusing on latency, packet loss, throughput and overhead generation.

Security study: ROS 2 performance with different QoS profile combinations for encrypted traffic data traffic, they measured for packet loss, latency, throughput, and overhead generation.

This research doesn't focus on giving suggestions for using ROS 2 in scenarios. Rather it presents a collection of performance metrics, across QoS and security configurations. The goal of this material is to support developers and researchers in enhancing ROS 2 performance helping them make decisions that consider cybersecurity requirements alongside the need for data transfer,

in autonomous robotic systems.

2.2.1 QoS Profiles

ROS 2 was designed to be used in a lossy network scenarios which utilizes User Datagram Protocol (UDP). UDP creates room for DDS to allow Quality of Services (QoS) policies that allows for fine-tuning data communication. Each data transaction is configurable at various levels of granularity that can effect our Throughput, Bandwidth, Redundancy, and Persistance. The following major policies are Liveliness, Lifespan, History, Depth, Reliability, Durability, and Deadline will be shown below.The ROS 2 version discussed in this paper is Crystal Clemmys includes the first three QoS policies that play a crucial role in customizing communication features, within a DDS setting. The others will also be described as well. Working together with DDS vendor Eprosima FastRTPS the specifics of these QoS policies have been outlined as follows:

HISTORY: This policy conserves message retention is divided into two approaches for handling sample or data storage. The 'KEEP ALL' method keeps all messages in memory regardless of quantity whereas the 'KEEP LAST' approach retains messages up, to a specified queue depth, which can be adjusted within DDS to meet specific storage needs [4].

Reliability: Two different strategies are used in this policy to ensure message delivery reliability. The 'BEST EFFORT' approach sends messages without needing confirmation from the recipient making delivery faster but risking message loss. In contrast the 'RELIABLE' approach waits for acknowledgments, from the receiver to minimize message loss even if it means a delivery speed [4].

Durability: This guideline outlines the way nodes handle messages that were sent before a subscriber joined a discussion. There are three approaches mentioned; 'VOLATILE' ignores any messages and only sends new information after subscription; 'TRANSIENT LOCAL' refills a new subscribers queue with previously saved messages on the local node; and 'TRANSIENT' gives a new subscriber access, to historical messages from storage ensuring that they have a complete history when they join [4].

Depth: Allows for the for a queue size to be activated for previous data samples only if 'history' is set to `keep_last` and will store the specified samples.

Lifespan: The maximum amount of time between the publishing and the receiving the message without the message being considered to be expired. Expired messages are dropped without

Case	*Participant*	*History*	*Depth*	*Reliability*	*Durability*
a	All	KEEP LAST	5	BEST EFFORT	VOLATILE
b	All	KEEP ALL	N/A	BEST EFFORT	TRANSIENT LOCAL
c	All	KEEP LAST	5	RELIABLE	VOLATILE
d	All	KEEP LAST	1000	RELIABLE	VOLATILE
e	All	KEEP ALL	N/A	RELIABLE	TRANSIENT LOCAL

Table 2.1: Quality of Service table used to pick out testing parameters in results from [19].

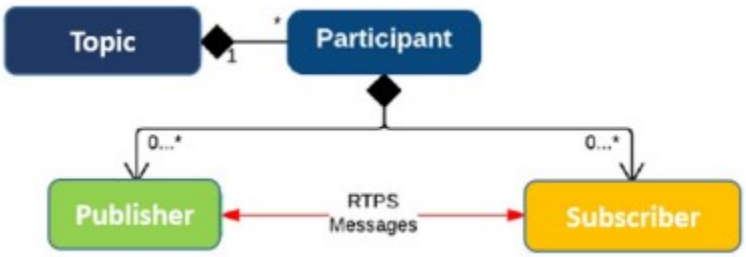

Figure 2.1: System setup with RTPS from [19].

logging of the drop [4].

Liveiness: Two different modes can be considered for this policy is which the first is the 'Automatic' which considers all of the nodes publishers to be alive for another "lease duration" when any of the publishers publish a message. Lease duration is the max period of time a publisher has to indicate that it is alive before it is considered down. The other is set 'manual' which sets which topics to which the publisher has to manually say it is still operational [4].

For the experiments and practical assessments, we used a basic ROS 2 network setup. This configuration, shown in Figure 2.1, consisted of a single topic with one participant, comprising both a publisher and a subscriber node. The tests made use of the standard ROS Middleware (RMW) called eProsima Fast RTPS to maintain consistency and relevance in line with commonly used ROS 2 setups. The researchers made sure to simplify the experimental setup to concentrate on how the QoS policies and security settings affected ROS 2's performance. This straightforward approach made it easier to link any performance differences directly to adjustments in QoS profiles and security setups, providing useful guidance for optimizing ROS 2 communication in various unmanned system scenarios.

2.2.2 Simulation Parameters and Results

When evaluating the strength and effectiveness of DDS security in the ROS 2 framework, they carefully outlined different simulation parameters to gauge how well the system performs. These parameters helped us grasp how the system behaves across various QoS profiles and security configurations.

Total Packets: This measurement captures the overall count of packets recorded by Wireshark, beginning with the first transmission of an RTPS message fragment and ending with the last one. It encompasses all associated information, like heartbeats (HBs), acknowledgment negation messages (ACKNACKs) and any messages related to the discovery protocol that come after the initial fragment.

Message (MSG) Fragment Packets: Fixed in size, these packets, counted within Wireshark, encompass solely the RTPS message fragments, excluding other forms of traffic or metadata.

Overhead Packets %: Represented as a percentage, this metric is the ratio of packets classified as overhead (those beyond the essential message fragments) to the total number of packets transmitted.

MSGs Lost: This is the tally of messages that the subscriber node failed to receive, which could stem from several issues such as lost fragments, entire messages going astray, data collisions, or other network anomalies.

MSG Fragment Latency (µs): The latency for each transmitted RTPS fragment is calculated by taking the difference in timestamps between the current message fragment and the preceding one.

MSG Latency (µs): The complete latency incurred in transmitting a single message is derived by summing the latencies of all RTPS fragments. This aggregate figure is then averaged over the totality of messages sent, which includes a consideration for retransmitted fragments and other messages that were not fully delivered.

MSG Throughput (Gbps): The throughput for each message is calculated by dividing the total bit size of the message by its latency in microseconds. The size of the message is the sum of all the constituent message fragments' sizes.

Δ %: : The throughput for each message is calculated by dividing the total bit size of the message by its latency in microseconds. The size of the message is the sum of all the constituent

message fragments' sizes.

The data tables and visual representations provided offer a detailed analysis of how different QoS settings and security setups affect DDS communication in the context of ROS 2. However, noteworthy findings from the simulation outcomes are:

1.) In instances where the file sizes of 0.25MB to 0.5MB, there was a notable difference in the loss of messages between unencrypted and encrypted scenarios, as encrypted cases encountered a surprisingly elevated rate of lost messages. 2.) As the sizes of files grew to 1MB and 2MB, there was a noticeable decrease in message loss during secure data transmission, suggesting that reliability in secure transmission varied based on the size. 3.) The delay went up for all file sizes especially when using consistent QoS settings indicating a balance, between data accuracy and delay. 4.) The throughput displayed a general decrease with increasing file size, which aligns with expectations considering the increased time required for larger file transmission.

The results clearly show that adding security features has an impact, on system performance than adjusting QoS settings. This finding is crucial for designers working on DDS networks those aiming for real time communication, in Cyber Physical Systems (CPS).

2.3 Router Design for DDS: Architechture and performance evaluation

The research paper referenced in [21] delves into an issue within the field of data distribution for real time systems for routing to multiple networks. The Data Distribution Service (DDS) for Real Time Systems is primarily tailored for communication within a domain, on a network. However this specific design orientation creates challenges when applications based on DDS aim to expand their communication capabilities to cover wide area networks (WANs). In cases limitations imposed by internet service providers (ISPs) on multicast and UDP traffic can present obstacles.

The research introduces a router model specifically created for DDS with the goal of preserving the protocols meaning while enabling data distribution, over wide area networks. This innovative design utilizes recovery methods and an overlay multicast approach to overcome the restrictions of unicast communication and ISP barriers. The brilliance of this proposed DDS router lies in its ability to improve system performance by managing messages and making use of resources. By giving priority to messages based on link expenses the router enhances speed. Maintains the QoS standards

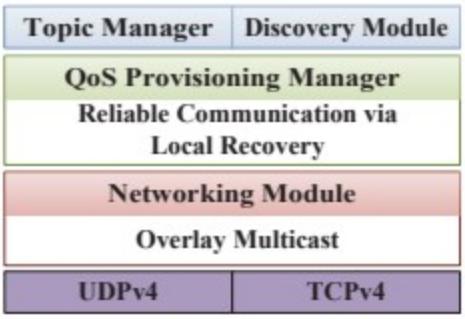

Figure 2.2: Question List Sent to Past Format Editors as Electronic Interview

to DDS.

By conducting simulations, with NS-3 the research offers real world proof of the effectiveness of the DDS router compared to traditional communication approaches. The findings show that integrating the DDS router significantly enhances system scalability and resilience which could have implications for implementing DDS, in wide area network settings. The contributions of this paper include:

1.) Introducing a cutting edge DDS router design that goes beyond the boundaries of network domains making data distribution, over WAN more effective, through the integration of recovery and overlay multicast techniques within the DDS framework.

2.) Through NS-3 simulations a thorough assessment showcases the effectiveness of systems that utilize the DDS router. This specifically emphasizes progress, in scalability and resilience compared to communication methods.

This study sets the foundation for future advancements in DDS networking, which could enhance the effectiveness and scope of real time data distribution systems. The findings of this research have broad implications across various sectors in IoT, industry, or military missions.

2.3.1 DDS Router Design

The unique design of the DDS router plays a role in addressing the challenges associated with distributing data across wide area networks (WAN). By preserving the aspects of the DDS and integrating features tailored for data distribution, over WAN the DDS router is able to streamline operations. This segment details the functions and structural elements of the DDS router.

Overview: The DDS router operates much, like a DDS participant integrating topic man-

agers and discovery modules for both the Participant Discovery Protocol (PDP) and Endpoint Discovery Protocol (EDP). It enables the transfer of topics over a WAN without compromising DDS semantics thanks, to its use of recovery and overlay multicast strategies. The design of the router not ensures message delivery. Also upholds time related QoS standards ensuring dependable communication.

Topic Manager: In charge of managing topics in the DDS router is responsible for handling topics from both distant domains. It's crucial for the DDS router to be aware of the topics, in networks and its local network to efficiently handle data distribution. This task is accomplished by utilizing the discovery module.

Discovery Module: Like DomainParticipants, in the DDS system the discovery module enables the DDS router to handle topics in various network areas. By creating domain participants, publishers and subscribers according to topic needs the DDS router guarantees that all routers and DDS entities share information, for topic publishing and subscription across the WAN.

QoS Provisioning Manager: When it comes to ensuring effective communication, the DDS router includes a local recovery plan to tackle problems like ACKNACK overload and too much TCP traffic. By functioning as a virtual publisher, the DDS router can recover and resend topic instances locally using ACKNACK messages, which helps make better use of network resources and decreases delays.

Overlay Multicast: To overcome the challenges presented by the inability to utilize UDP/IP multicast over an area network the DDS router utilizes an overlay multicast approach. It establishes a multicast tree using link costs facilitating data transmission routes, throughout the network. This technique notably minimizes the need for TCP connections and session messages providing an advantage, for real-time applications that depend on rapid data distribution.

Figures 2.3 and 2.4 show how the recovery and overlay multicast strategies are put into practice. In Figure 2.3 you can see the steps involved in recovery starting from sending topic instances to aggregating and re-transmitting ACKNACK messages. Figure 2.4 illustrates how the overlay multicast tree is created and the paths, for transmission explaining how the DDS router improves data flow, between network domains.

The design of the DDS router showcases how the DDS standard has been creatively applied to tackle networking obstacles. Through incorporating these features the router not only boosts DDS effectiveness in WAN setups but also paves the way for improved simultaneous data sharing,

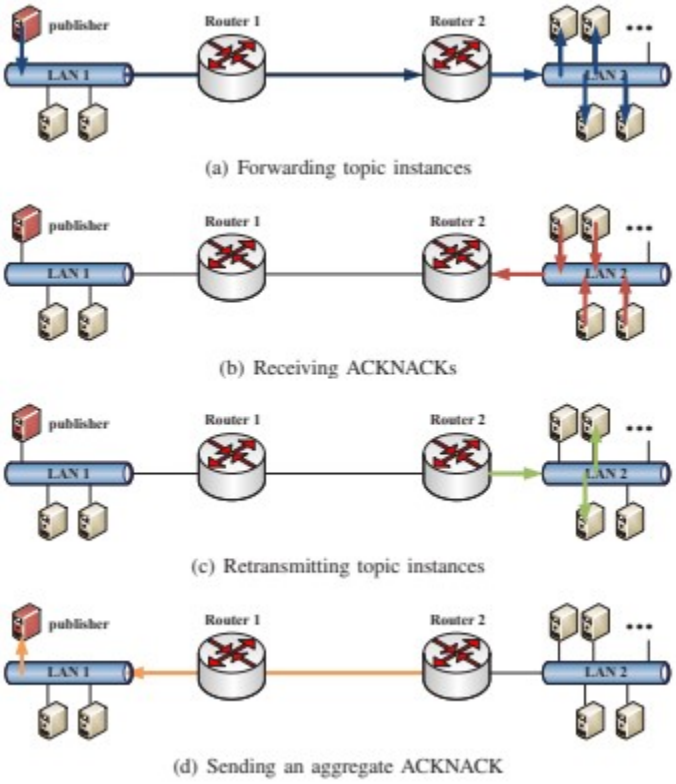

Figure 2.3: Local Recovery Process

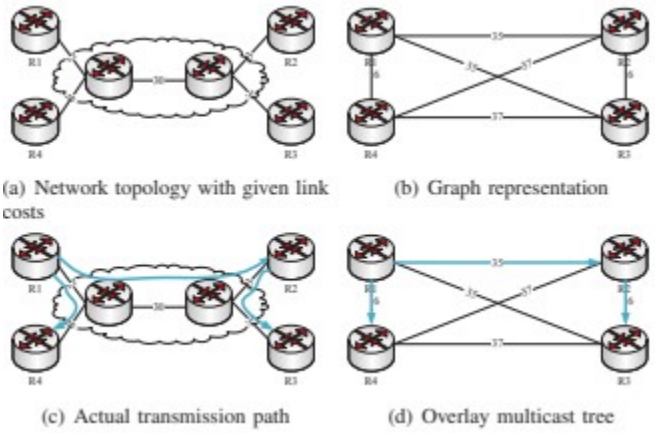

Figure 2.4: Overlay Multicast Strategy

across geographically distant networks.

This thorough examination does not tackles the requirement for strong security measures in multi-hop DDS networks but also adds to the ongoing conversation about improving DDS and ROS 2 systems for real-time critical applications. By assessing how QoS and security interact within these frameworks we strive to provide insights and approaches that can enhance the deployment of DDS networks ensuring their effectiveness and dependability on working towards evolving cybersecurity challenges.

Our research aims to bridge this gap by exploring the setup of DDS networks and analyzing how transmitting types of data, with active security protocols impacts a multi-hop network. Additionally we plan to simulate data flow in a networking testbed environment to understand the performance capabilities of DDS networks particularly in scenarios resembling battlefield conditions. This method intends to assess the viability of DDS systems, as real time Cyber Physical Systems (CPS) in security conscious settings.

Chapter 3

Multi-Hop SROS Analysis

This chapter show our unique design approach for DDS Routing in Secure ROS 2 robot fleets and the implications of implementing it across various networks and seeing the impacts of multi-hopping. Our study is structured to cover a range of perspectives with a focus on ensuring packet transmissions between publishers and subscribers in DDS's communication protocols. We examine the reliability of these transmissions while maintaining the security of packets as they travel through networks. Our approach involves analyzing network performance, particularly looking at subscriber throughput and latency when different payload sizes are sent by the publisher. This analysis is crucial for understanding how payload sizes affect network behavior, such as node responsiveness and message handling efficiency. Through capture analysis, we investigate how message fragmentation occurs with varying payload sizes in both LAN and WAN.

The experiments are based on a network setup that mimics real-time ROS 2 configurations. This simulation allows us to adjust variables such as payload sizes and observe their impacts in a controlled yet scalable environment, giving us insights supported by empirical evidence instead of just theory. After providing an overview of the architecture, we will showcase the data collected from our experiments using the graphs and tables presented below. These visual aids will demonstrate how network performance metrics interact with payload characteristics, giving a picture of the patterns and anomalies observed during the tests.

Following that, we will analyze the results to uncover the reasons behind phenomena such as packet loss, latency spikes, or fluctuations in throughput. Our thorough assessment aims to highlight current limitations and obstacles and spark conversations about potential improvements.

We intend to propose strategies for experiments that could enhance the DDS Routing process, strengthen network security protocols, and streamline data transmission in robot fleets operating across large-scale LAN and WAN setups.

3.1 Network Topology and Set-up

Our test environment is designed to support the communication between ROS 2 robot fleets across LANs. Key to this setup are the basestations, which act as nodes responsible for facilitating message exchange between networks. Each basestation is connected to its LAN and is also linked with one or more peer basestations creating a network that connects all individual fleets. This mesh network layout is illustrated in the diagram offering a visual representation of the network structure.

To overcome the limitation of DDS's inability to multicast across networks for the discovery protocol, we developed a ROS 2 repeater application that will be the middle man between the wanted networks. This application subscribes to topics within its LAN and republishes them, ensuring that the original data packet integrity from the initial publisher is maintained. This innovative solution utilizes DDS's capabilities within LANs and demonstrates our system's adaptability in handling various data types required for inter-network communication.

In order to measure and evaluate data transmission accuracy we have integrated logging tools into each ROS 2 node. These tools record timestamps indicating when a packet is sent by the publisher and successfully received and reassembled by the subscriber allowing us to obtain latency measurements and processing time for the repeaters. We also keep track of the data received by each node to calculate the average speed at the end of the test. Moreover, we carefully log the transmission of packets, keeping track of how many are sent out by the sender and received by each recipient. These measurements show that they play a crucial role in assessing how effective and dependable our message-handling system is across large LAN/WAN networks. Each experiment will show the throughput, latency, packet success rate, and processing observed for one, two, and three network hops, correspondingly. The intention behind incrementing the number of hops is to simulate the practical scenarios where data must traverse multiple network segments, such as in distributed robotic systems applications that span across different geographical locations. The next part will discuss in detail the network simulator we used for our experiments, explaining how it mimics real-world networking conditions and contributes to the strength of our analysis.

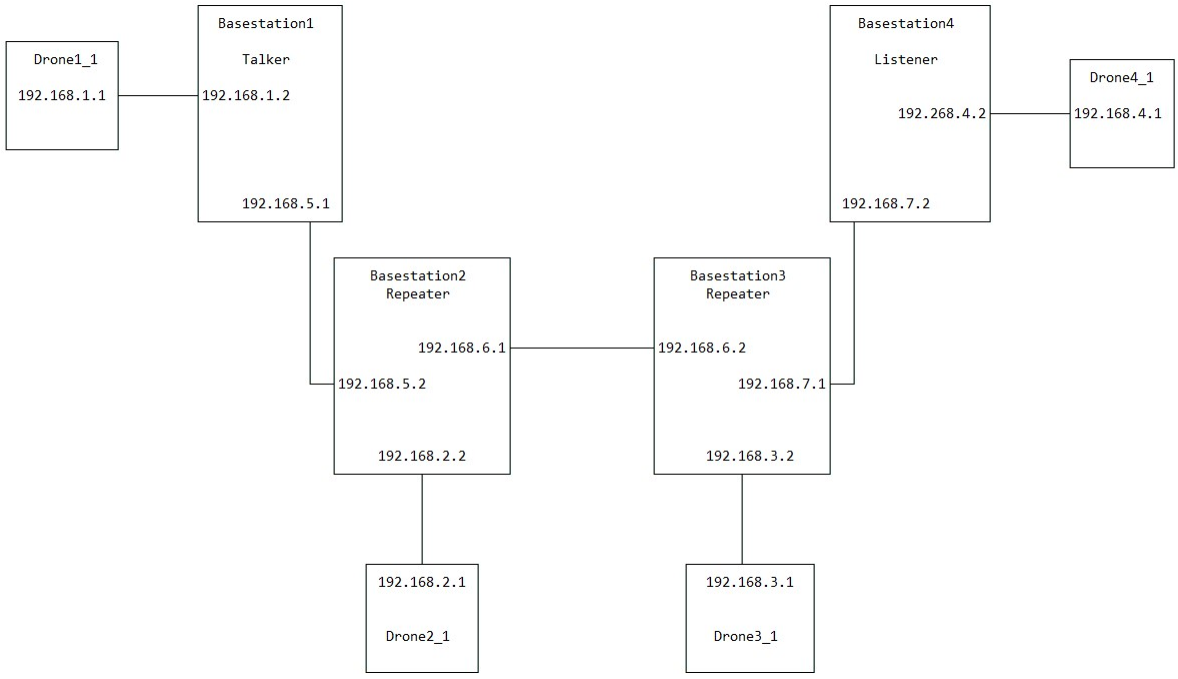

Figure 3.1: Extended-LAN for testing Multi-Hop performance.

3.1.1 Network Simulator: Fabric

This experiment was conducted on the FABRIC Testbed [6], a National Science Foundation (NSF) funded research project. FABRIC is a "testbed-of-testbeds" [35] for computer network research, providing researchers sandboxed access to an international network built with cutting-edge network infrastructure and hardware. FABRIC is interconnected with many other popular research testbeds and networks such as Chameleon Cloud, CloudLab, the PAWR testbed, and Internet2 [6], allowing researchers to instantiate experiments on each testbed and measure the network performance of transmitting data between testbeds and between nodes on the FABRIC network. Such flexibility allows researchers the ability to create and experiment with novel network protocols, routing, and cybersecurity techniques. FABRIC also provides a Measurement Framework (MF), allowing network performance to be monitored and analyzed in real-time using dashboard tools such as Grafana and Kibana. Although neither the ability to connect to other testbeds nor using FABRIC's MF to analyze the network were implemented in our work presented in this paper, we expect to use these tools in the future.

FABRIC provides users with a JupyterHub account, where users can create and store

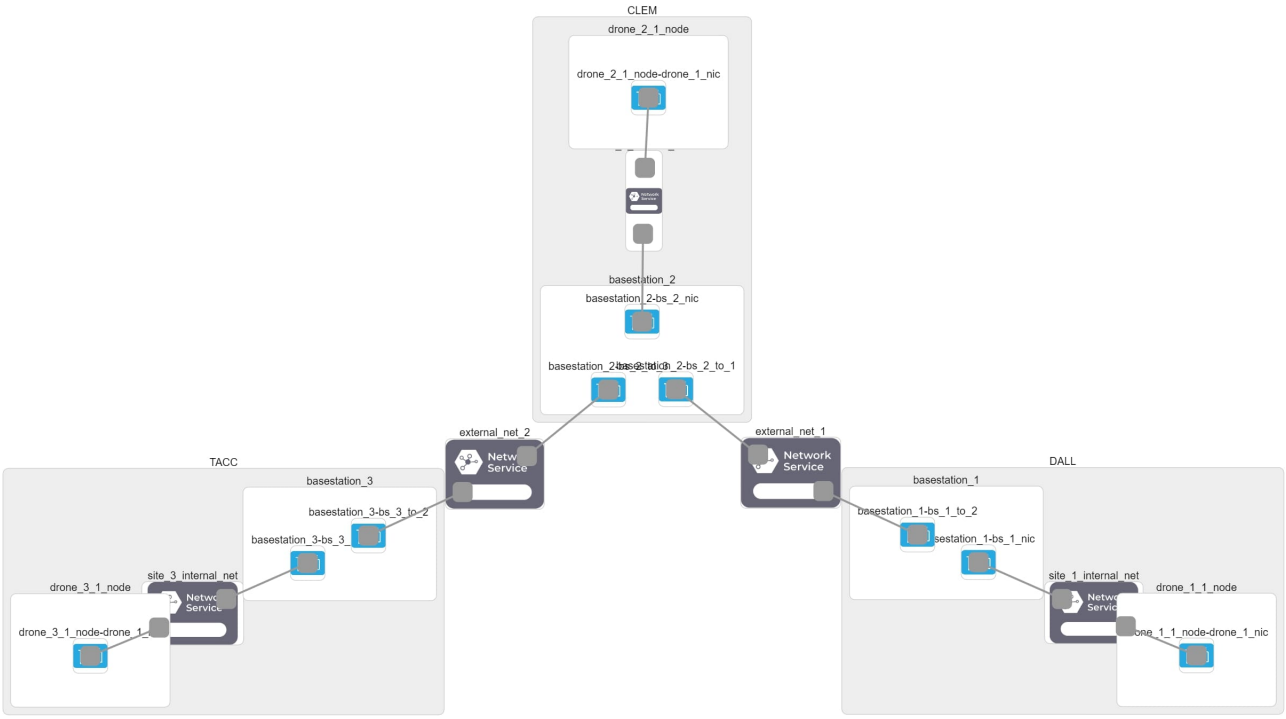

Figure 3.2: Example Image of testing topology.

Python-based Jupyter Notebooks. Using these notebooks, users can instantiate FABRIC experiments called slices, which contain any number of nodes (virtual machines) across any number of FABRIC sites (the locations of servers connected to the FABRIC network). Users can choose how nodes are networked; FABRIC provides both layer 2 (L2) and layer 3 (L3) connections. Many flavors of operating systems are provided for the user to choose from for each node.

3.2 ROS 2 security results

In this detailed section, we outline the results of our networking tests that were carefully planned to assess the performance of our DDS routing architecture in extended LAN/WAN setups. The results presented here form a solid basis for evaluating the speed and delay across a daisy-chained network of ROS 2 robot fleets and their corresponding basestations.

Our analysis begins by looking at the speed across various payload sizes, ranging from 1K to 1M. These payloads are gradually increased to mimic different operational needs and travel through multiple hops each leading to a specific network or efficiently relayed by our custom ROS 1-3 re-

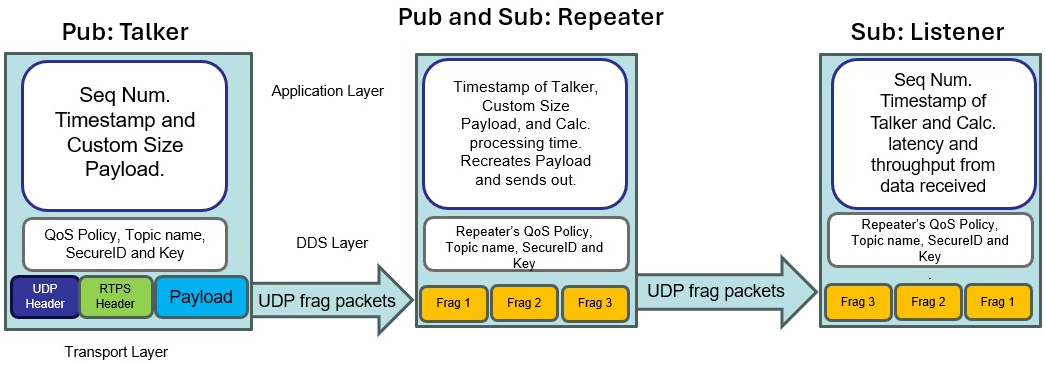

Figure 3.3: Example Image of testing topology.

peaters. These repeaters play a crucial role in our experiment setup by facilitating the smooth transfer of data packets across different network areas. To avoid potential congestion points and ensure the reliability of our data gathering process, each publisher in our network is set to send messages at timed intervals of 500 milliseconds. This timing is chosen deliberately to prevent overwhelming the publisher's sender buffer, which could result in artificial delays or packet loss that might affect our findings.

The experiments are not only aimed at measuring speed but also at observing how payload size and hopping through repeaters impact packet delivery intricately. This involves closely examining the time it takes for data packets to move through each connection point and how it affects the overall efficiency of communication.

After looking at the data transfer rate, we shift our focus to analyzing the success rates of packet deliveries. This means diving into the percentages of successfully delivered packets across different sizes. By breaking down these percentages, we can understand how well our network performs under varying data loads. We also assess how well the DDS routing system handles increased data demands and whether the repeater nodes effectively maintain accurate data transmission.

In addition to these numerical findings, we engage in a qualitative conversation about how the network behaves in reality. This covers potential reasons for lost packets, irregularities in data transfer rates, and fluctuations in latency, offering a comprehensive view of how well the network functions. We also suggest possible explanations for observed trends in the data and provide a detailed discussion on what they mean for future improvements to network designs.

Overall, this section evaluates our DDS routing approach, paving the way for subsequent

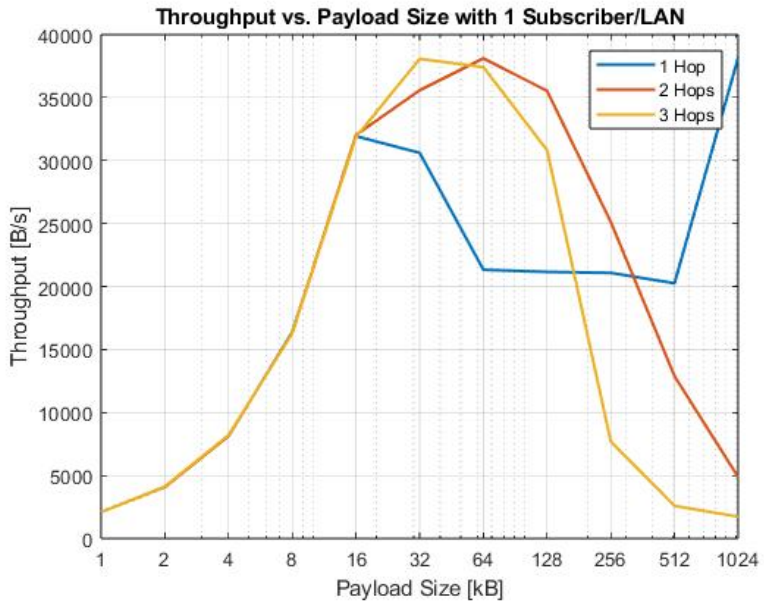

Figure 3.4: The throughput analysis for each Hop with 1 Subscriber with varying payloads.

sections that will delve deeper into specific aspects of network performance. As we move forward, we aim to draw meaningful insights from the data that can offer valuable perspectives on enhancing communication systems for networked robots.

3.2.1 Throughput for Payload sizes each Hop

Our research thoroughly explores how efficiently ROS 2 publishers transfer data through network connections to assess the reliability of secure data transmission. This analysis is vital for understanding if the DDS routing system, which is meant for safe communication performs well when sending messages of different sizes between network points. The graphs shown visually represent the data transfer rates achieved for payload sizes ranging from 1K to 1M bytes giving us insights into how network scalability impacts our ROS 2 setups robustness. Our experiments involved sending messages from ROS 2 nodes with increasing payloads to mimic real-world data loads. To prevent overwhelming the publisher's buffer and causing loss or delays, we sent messages at regular intervals of 500 milliseconds. This approach allowed us to evaluate how well the network handled packets without disrupting the flow of information.

Interestingly, we observed a decrease in data transfer speeds across all tests once the payload

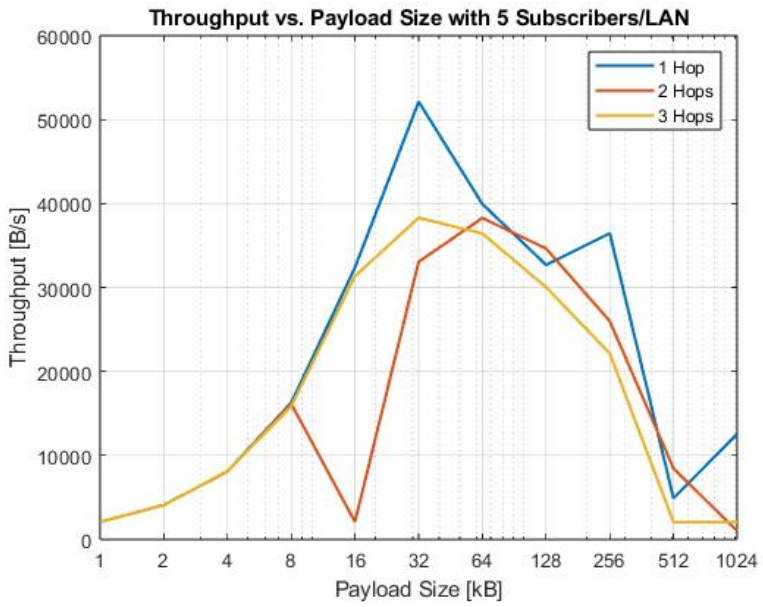

Figure 3.5: The throughput analysis for each Hop with 5 Subscribers with varying payloads.

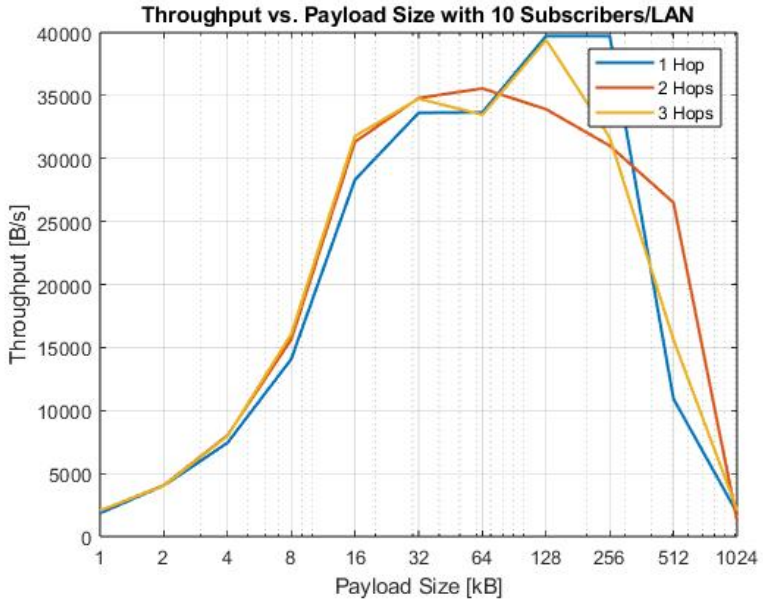

Figure 3.6: The throughput analysis for each Hop with 10 Subscribers with varying payloads.

size reached 64kB, falling within a bandwidth range of 35,000 to 40,000 bytes per second or lower. This consistent pattern across network hops suggests a potential limit in the DDS routing system's capacity based on our current settings. Upon examination, it appears that smaller data loads move smoothly through the network, maintaining high speeds. However, when larger data packets are transmitted, a decrease in performance is observed. This decrease becomes more noticeable as the number of network connections increases highlighting the importance of having a network structure or optimizing how messages are queued and managed.

Moving forward, our goal is to investigate why this performance degradation is happening. We will analyze factors like network congestion, buffer capacities, and how well the transport protocols are working. By gaining insights into these aspects, we aim to suggest improvements that can enhance performance levels and ensure data transmission across LAN/WAN networks over longer distances regardless of payload sizes.

3.2.2 Latency for Payload sizes each Hop.

In our second section, we introduced the latency analysis by the DDS routing/repeater mechanism across multiple network hops; we directed our attention to the delays encountered by messages as they transfer messages across each extended network. The statistics of the ROS 2 publishers' performance, in relation to the number of repeaters and the size of the payloads, are visualized in the series of graphs presented. A discernible pattern emerges, with latency maintaining a general incline as payload sizes increase, only to experience a steep surge beyond the 128kB mark. Throughout all scenarios, we can see a pattern from just one user in the system to more crowded environments with five or ten users. The graphs clearly show how latency changes for each of these setups and point out an increase in latency when dealing with larger data packets.

For 1 Subscriber/LAN: The graph demonstrates a rise in latency up to a payload size of 16kB. However, once we surpass the 128kB mark, there is a significant spike in latency indicating potential network architecture or DDS configuration challenges when handling larger data loads.

For 5 Subscribers/LAN: The trend mirrors what we observed with one subscriber. The spike in latency is more pronounced. This suggests that having more subscribers could impact the network's efficiency in processing payloads.

For 10 Subscribers/LAN: This scenario further confirms the trend as latency increases more significantly with a higher number of subscribers. It shows how network scale directly influences

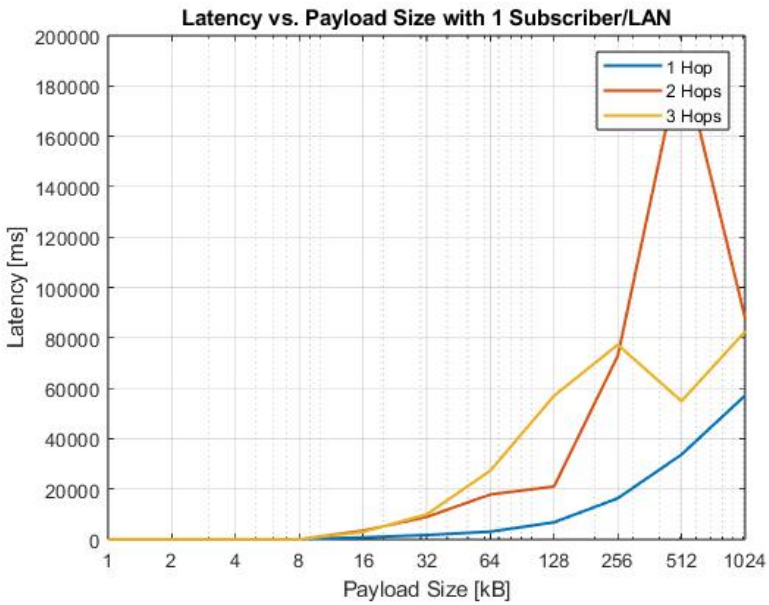

Figure 3.7: The latency analysis for each Hop with 1 Subscriber with varying payloads.

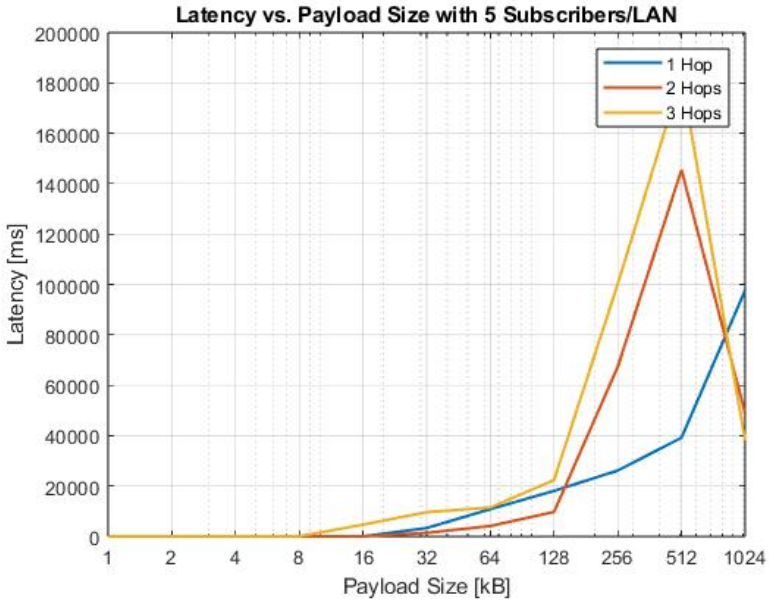

Figure 3.8: The latency analysis for each Hop with 5 Subscribers with varying payloads.

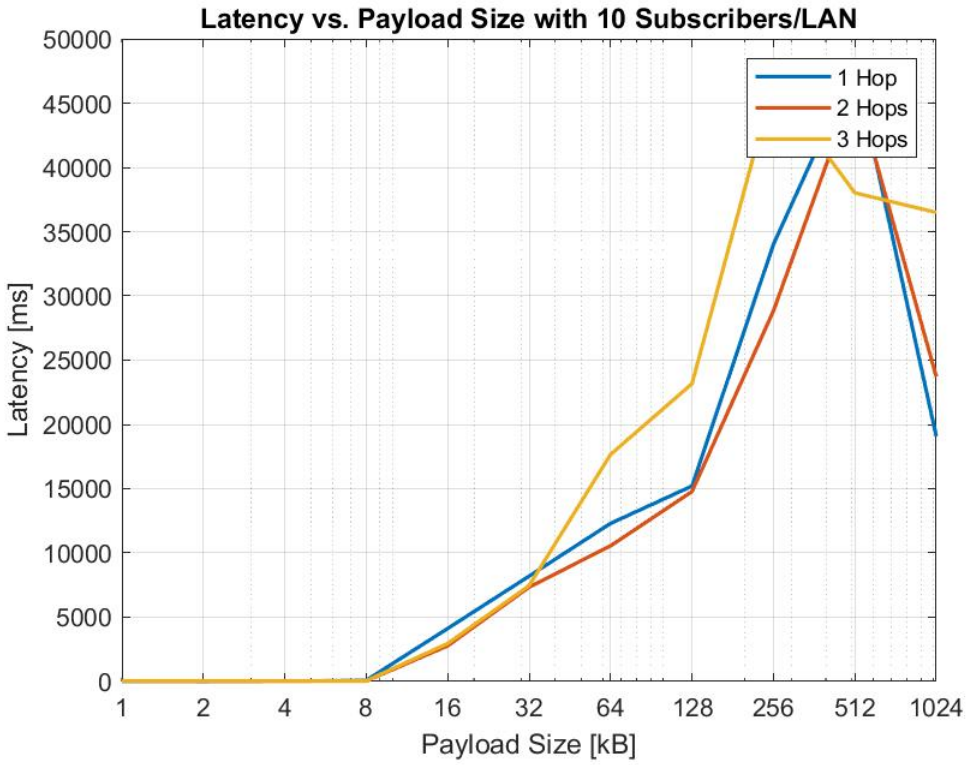

Figure 3.9: The latency analysis for each Hop with 10 Subscribers with varying payloads.

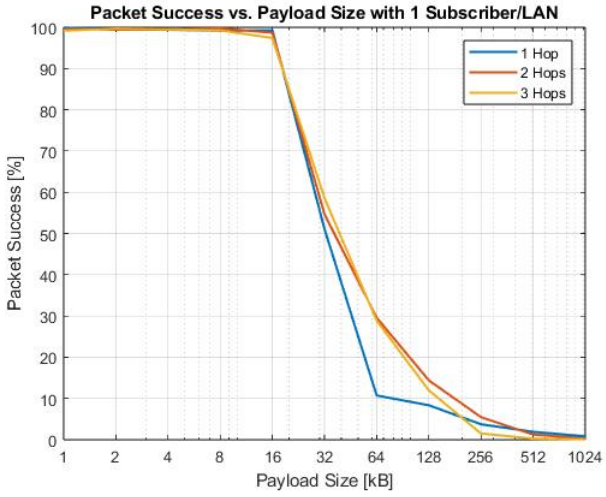

Figure 3.10: Packet transmission success rate for each Hop with 1 Subscriber with varying payloads.

latency.

These patterns in latency provide insights into how payload size, network connections, and subscriber count interplay within the ROS 2 communication framework. As the amount of data being transmitted increases, the network's response time changes significantly, showing how dealing with and delivering large data over a wider LAN/WAN infrastructure becomes more challenging.

Although the response times for amounts of data initially meet the requirements for real-time tasks the significant increase in response times for larger amounts raises important questions about how well DDS routing can handle heavy data loads. This finding is crucial for shaping research and improvement efforts to reduce sudden spikes in response times and improve the overall effectiveness of DDS based communication in large-scale and distributed systems.

3.2.3 Packet Success rate for Payload sizes each Hop.

Our examination of success rates plays a crucial role in assessing how messages are reliably transmitted within a DDS based system. In this study, we measured the success rates of packets sent from the sender to the receiver across message sizes ranging from small 1kB messages to larger 1MB loads. The accompanying visuals illustrate how success rates change with the number of network hops emphasizing how larger payloads and network intricacies affect data transmission reliability.

The data shows a trend that success rates remain consistently high for smaller message sizes

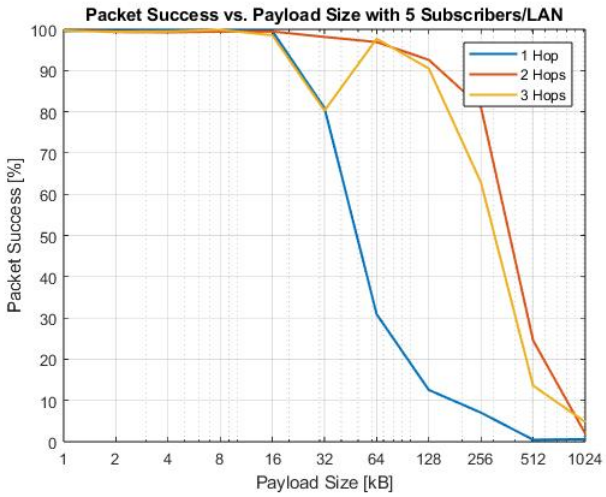

Figure 3.11: Packet transmission success rate for each Hop with 5 Subscriber with varying payloads.

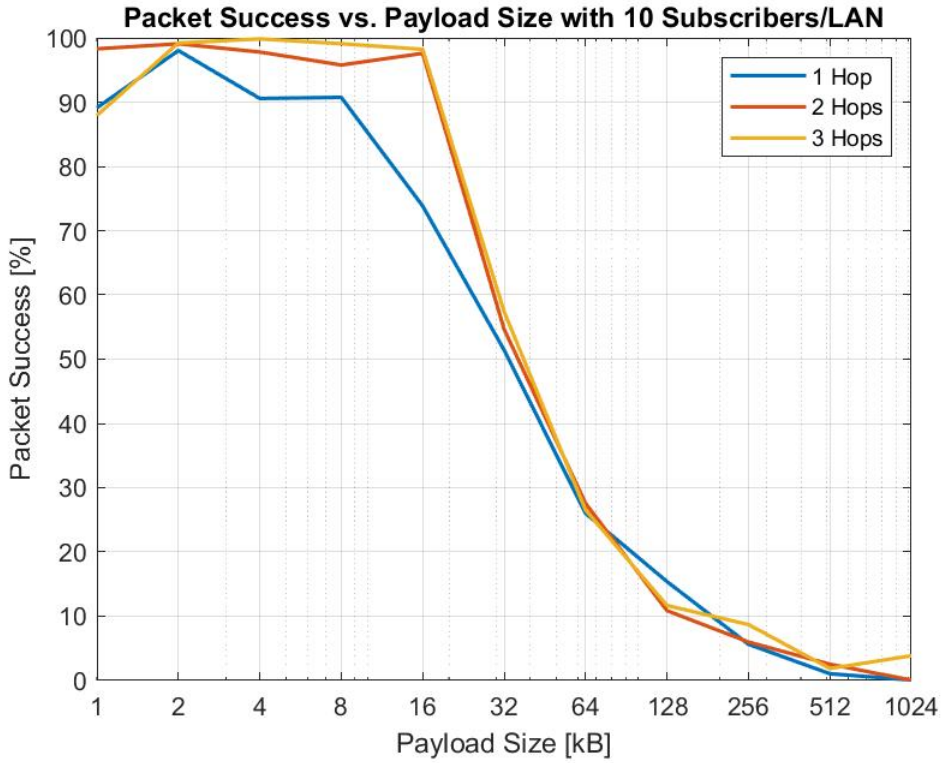

Figure 3.12: Packet transmission success rate for each Hop with 10 Subscribers with varying payloads.

regardless of the number of hops. However, as we surpass the 16kB threshold, we start to see a drop in success rates, especially with larger payloads.

For setups with 1 Subscriber/LAN, smaller payloads have perfect success rates regardless of hop count. As payload sizes increase, success rates decrease noticeably, particularly in the 64kB to 512kB range. For setups with 5 Subscribers/LAN, a similar pattern emerges. The dropoff is more abrupt. Multiple hops further amplify this decline, indicating challenges in handling larger payloads across various network segments. With 10 Subscribers/LAN, the impact of payload sizes becomes more evident as the success rates decrease significantly as the payloads increase. The scenario involving 3 hops shows a notable effect on transmission reliability.

The relationship between the number of hops, payload size, and packet success rates highlights the difficulties in maintaining communication across extensive networks. The consistently high success rates for messages indicate an efficient system under lighter workloads. However, the significant drop in success rates for messages suggests strain on potential limitations in the current DDS transport setup.

This observed pattern calls for an examination of network resource allocation buffer sizes, message queue management, and transport protocols to pinpoint and address packet loss causes. These improvements are essential for applications where large payloads and extensive networks across expansive robotic operations.

Recognizing and resolving these challenges is crucial, for advancing DDS-based systems to ensure they can effectively handle the demands of modern communication networks. These insights will inform optimizations that enhance DDS's resilience and effectiveness in meeting increasing data needs.

3.2.4 Processing times for Repeaters

This section goes over the processing duration at the repeater nodes. We calculated the mean sum of the processing times for each repeater, which reflects the interval required to accept a published message, strip its original topic header, and then broadcast it under a new topic with the corresponding payload and timestamp from the original publisher. The provided figures highlight a stark escalation in processing times correlating with increasing payload sizes, potentially creating significant delays within the system. As we introduce additional repeaters, the cumulative processing time extends further due to each repeater's need to modify the topic header for its respective network.

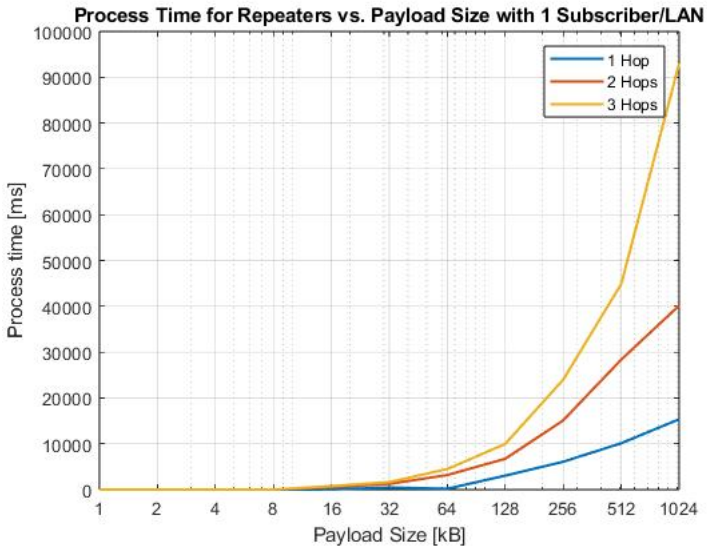

Figure 3.13: Latency analysis of 1 Subscriber comparing different QoS policies.

These findings point to a considerable processing bottleneck impacting the efficiency of our ROS 2 nodes.

3.2.5 Discussions of Results:

The extensive tests we conducted in our network setup provided an insight into performance concerning data size. While our testing setup with hops did cause some delays, the network resilience stood out, especially considering the security measures like key exchange authentication that had minimal impact on the overall performance of the ROS nodes. However, when it came to transmitting data loads, we noticed a significant impact on network efficiency. This highlights limitations within the Fast DDS system, particularly its default message size limit of 64kB which aligns with standard TCP/UDP payload sizes [13].

The reasons behind the observed drop in performance seem to stem from two issues. Firstly there's a concern about network packet loss due to socket buffer overflow. When the amount of data being transmitted reaches capacity and saturates the buffer packets are dropped, creating a bottleneck. This is most noticeable at the repeater node, where the buffer has to handle a surge of RTPS messages from the publisher node, resulting in decreased throughput and success rate.

Secondly, as data sizes surpass the Maximum Transmission Unit (MTU) of the underlying

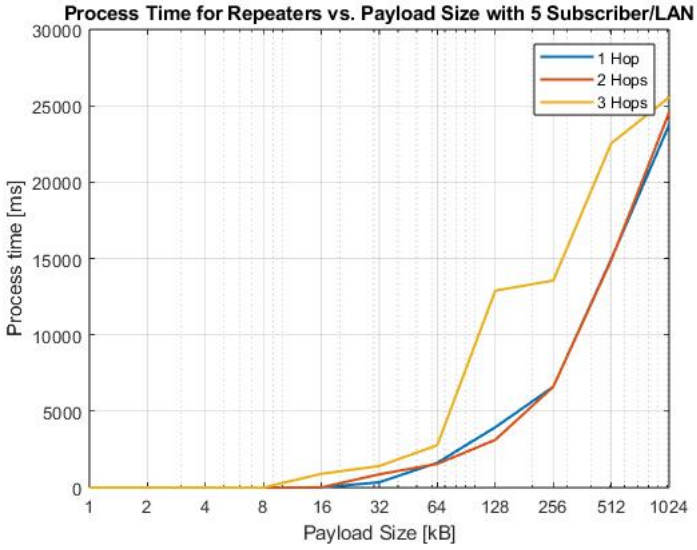

Figure 3.14: Latency analysis of 1 Subscriber comparing different QoS policies.

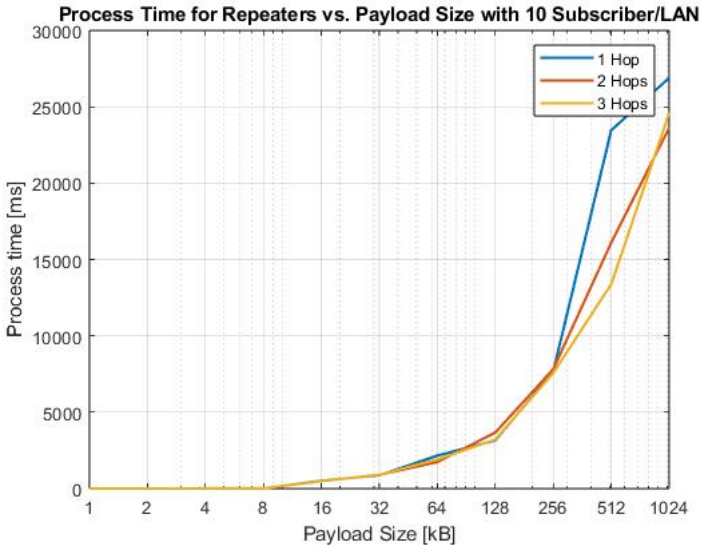

Figure 3.15: Latency analysis of 1 Subscriber comparing different QoS policies.

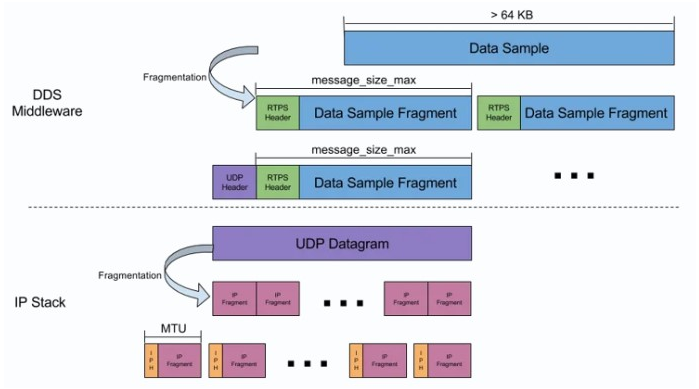

Figure 3.16: UDP fragmentation on the DDS level from [34].

network IP level fragmentation becomes an issue. When data gets broken up, the system uses a buffer to keep the pieces until the entire message can be put back together into UDP datagrams so the DDS level can process [30]. If this process is not met in the timeframe set in the QoS policy 'lifespan' then the message will be dropped [4]. Also, this process of putting things back is both demanding on resources and fragile; and if an IP fragment is lost, then the NIC cannot reassemble the UDP packet, and the DDS payload will be lost[13] [34].

This vulnerability to losing packets is clearly seen in our test results. As we increased the payload sizes there was a drop in successful packet transmission rates. This pattern aligns with the idea that larger payloads are more likely to break into fragments and face the risk of losing those fragments.

Based on these findings it appears that DDS based systems may need two strategies to maintain high performance when dealing with large amounts of data especially in secure scenarios. On one hand adjustments, in the DDS setup should be considered, like increasing socket buffer size and tuning message size settings. On the hand effective management of IP level fragmentation is crucial; this could involve using advanced QoS mechanisms to ensure that large fragmented payloads are delivered intact and reliably.

In summary, this research highlights the significance of optimizing systems when expanding DDS-based communication networks. Tackling the issues of buffer control and IP fragmentation is essential to guaranteeing the reliability of DDS systems in managing data loads, especially crucial in intricate distributed systems across multiple networks.

3.3 Fine Tuning DDS Transmitting Large Data Loads in ROS 2

In the section, we found that our network setup, with standard Quality of Service (QoS) rules and security measures, effectively handles data packets of up to 32kB. However, this setup struggles when dealing with expanded subscriber memory buffers, resulting in more than optimal performance. This situation calls for a transport protocol to ensure smooth packet delivery across network stages.

We focus on improving our data transmission approach to manage packet sizes more efficiently. We suggest a customized DDS QoS setup aimed at enlarging the buffer and switching our transport protocol from UDP to TCP. This change is intended to enhance delivery reliability and reduce delays associated with larger data loads.

Additionally, we plan to adjust the message sizes at the DDS layer as a proactive step to avoid IP fragmentation issues that can arise when payloads surpass the MTU limit, especially in Ethernet networks, while also taking in mind the security overhead for RTPS must included for each packet. By tweaking these settings our goal is to establish a robust and effective network structure capable of meeting the demands of modern distributed systems where handling large data loads is increasingly common.

3.3.1 Experiment setup

In our efforts to strengthen our network's ability to manage extensive data transfers, we have refined our experimental configuration to concentrate solely on optimizing the transmission of large payloads. Similar to setups, our network spans across multiple points, but this time around, the key focus is on skillfully applying Quality of Service (QoS) policies and utilizing strategic configurations derived from FastDDS capabilities for handling large data exchanges [14].

In this test run, we go into fine-tuning the QoS parameters by carefully tailoring each ROS 2 node's policies to ensure smooth operational performance. We lost the first couple of messages by selecting a `best_effort` durability setting to streamline processes within the somewhat slower TCP transport but decreases message delay. Additionally, we set a ten-second deadline for message delivery to prevent messages from being dropped due to potential buffer congestion.

To improve how the transport layer manages data loads, we activated Fast DDSs `large_data`

mode, a feature that provides various adjustments aimed at facilitating the seamless transfer of sizable datasets. This mode allows for flow control between an asynchronous publisher and its subscribers to avoid traffic congestion at the read and write buffers [15][30]. This mode plays a role, in overcoming standard message size limitations and steering clear of fragmentation issues that often arise when handling robust data streams.

We started by adjusting the `max_msg_size` parameter from 1MB to 2MB to avoid message fragmentation and ensure that complete messages fit within this size limit. At the time, we increased the `socket_size` first to 2MB and then to 4MB making sure that this buffer could handle messages larger than the maximum size thus preventing buffer overflow. Additionally, we enabled blocking sockets by setting the `non_blocking` parameter to true in our setup. This decision was intentional; although it does pose a risk of losing messages when buffers are full, it helps maintain data flow without causing the application to hang, prioritizing continuous data transfer over absolute reliability [14]. Lastly, we established a `tcp_negotiation_timeout` of 50 seconds. This setting ensures that the logical port is ready before starting data transmission reducing the chances of losing messages during the negotiation phase. However, this extended timeout may lead to delays in the discovery process while enhancing reliability, when `best_effort` reliability is active.

By increasing message and socket buffer sizes and implementing blocking sockets our configuration is optimized to take advantage of TCP transport capabilities. This setup aims to provide a transmission experience similar to streaming videos, where seamless delivery is key. In making tweaks to our DDS configuration, we aim to shape a network structure that can not just handle but excel in meeting the challenges posed by high-volume data transmission requirements.

3.3.2 Latency analysis

The experiment to optimize network robustness while maintaining security integrity often leads to an intricate balance, particularly when scaling payload sizes. Our recent experiments delved into the latency implications of transmitting increasingly larger payloads across a ROS 2 network using TCP. This setup, inherently demanding Acknowledgments (ACKs) from subscribers, unveiled a tangible deceleration in message delivery times, spotlighting the trade-offs when prioritizing secure and reliable data transmission.

Figures 3.17 and 3.18 show the latency disparities across scenarios with one subscriber per LAN and ten subscribers per LAN, respectively. The experiments, designed to explore the bounds

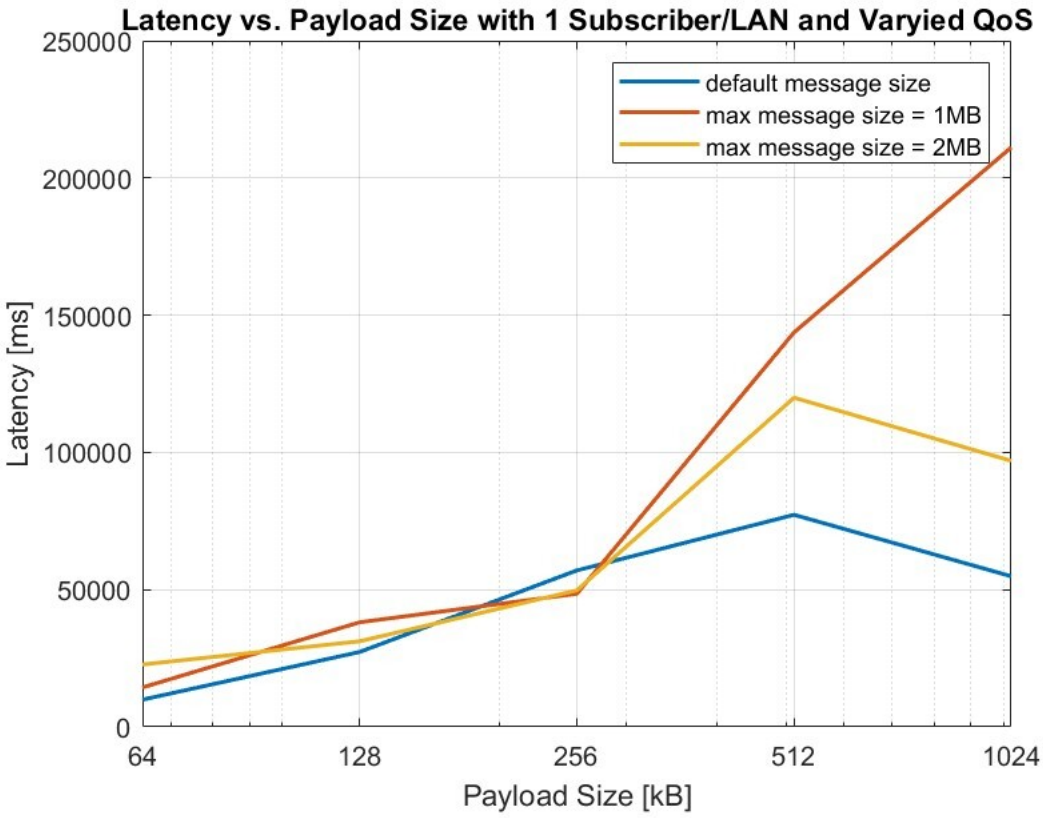

Figure 3.17: Latency analysis of 1 Subscriber comparing different QoS policies.

of Fast DDS's `large_data` mode, show latency landscape as payloads expand from 64kB to 1MB. This analysis reveals a pronounced latency increase, especially at payload thresholds where TCP's reliability checks—in the form of ACKs introduce a significant overhead. The result of the delay scales in relation to the message size, with the largest payloads incurring up to fourfold increases in latency. This observation underscores the inherent latency cost that comes with the assurance of TCP's delivery guarantees, a cost that becomes more acute as the payload size crosses certain thresholds.

With a single subscriber, the system maintains a relatively stable latency profile until payloads exceed the 256kB mark. Beyond this point, we observe a steep incline in latency, reflective of the system's strain under heftier data burdens. As we introduce more subscribers, the latency impact magnifies, as evidenced by the inflection points in Figure 3.14.

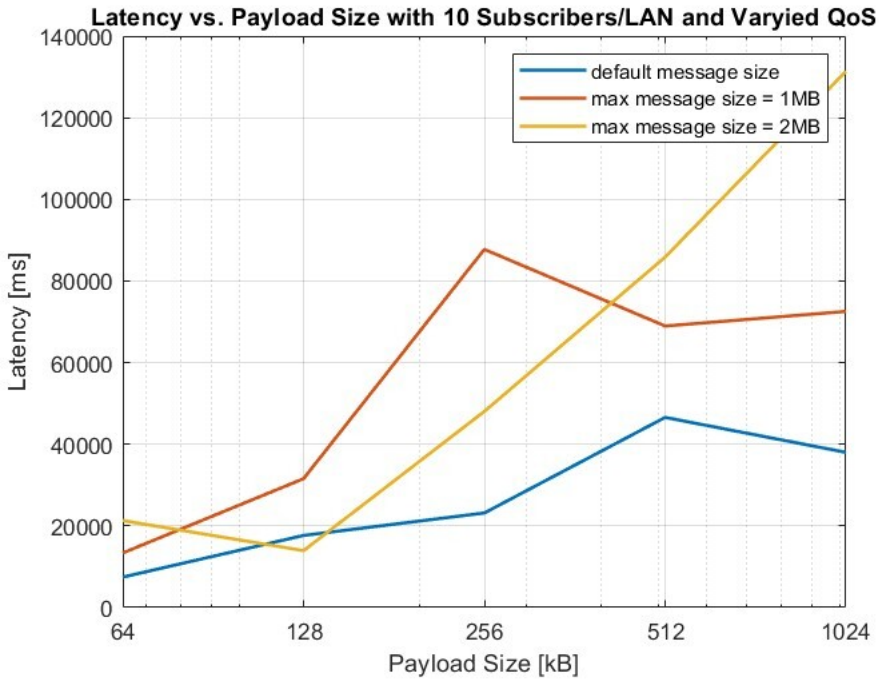

Figure 3.18: Latency analysis of 10 Subscribers comparing different QoS policies.

3.3.3 Throughput analysis

In our efforts to enhance the communication setup in ROS 2 for handling data transfers, we concentrated on boosting the network node's capacity for throughput. The test results indicate an enhancement in maintaining throughput even when we increased the testing data sizes to 1 MB. This can be seen in Figures 3.19 and 3.20, where the throughput with default message sizes drops below 10 kB/s after reaching the 512 kB payload threshold. On the hand using `large_data` modes has proved effective in keeping a steady throughput of around 30 kB/s even with hefty 1 MB payloads.

The significant stability in performance achieved through employing `large_data` modes can be mainly attributed to their ability to prevent overwhelming the repeaters receive buffer. When this buffer gets overloaded it results in dropped packets, a challenge effectively handled by using `large_data` configurations. As a result, it ensures that our 'deadline `best_effort` QoS policy is maintained intact guaranteeing dependable throughputs throughout the network. This crucial aspect of network performance highlights the usefulness of

Chapter 4

Conclusions and Discussion

This chapter summarizes the achievements and core objectives outlined in this thesis. The most important was the analysis of ROS 2 security performance across varied network domains, with the ultimate goal of developing a network architecture reflective of real-world autonomous system applications. Leveraging the inherent DDS framework, we established interconnected network segments, facilitating seamless communication and data relay across active ports within the FABRIC testbed environment. This strategic setup enabled a critical evaluation of ROS's network performance under a spectrum of payload conditions.

Custom ROS nodes were crafted and deployed, serving as testbeds to evaluate and refine Quality of Service (QoS) policies, thereby enhancing network efficiency. The investigation further delved into the limitations posed by ROS 2's default UDP protocol when handling substantial payload sizes, leading to the discovery and implementation of alternatives that markedly bolstered network robustness. The enhancements were tested against different scenarios, simulating conditions in which ROS 2 mechanisms/networks that would be best fitted for battlefield conditions, thus ensuring the developed system's reliability and readiness for high-stakes deployment.

4.1 ROS 2 Security Considerations

The results of our first batch of testing revealed the effects between payload sizes and network performance for ROS 2 systems, particularly the limitation of the 64kB and above which is inherent to UDP datagrams. This limit became a pivotal point, above which the data fragmentation

due to Ethernet's MTU limitations precipitated a marked degradation in throughput, latency, and packet success rates. Considering this reason, we address the bottlenecks by being able to handle payload sizes bigger than 64kB by implementing `large_data` mode that uses TCP. This enhancement showcases its proficiency in ensuring data integrity through improved flow control mechanisms. This switch to TCP effectively mitigated the packet loss and maintained consistent throughput across varied payload sizes. However, it introduced increased latency, as the protocol's dependence on acknowledgment receipts from subscribers inherently slows down the data transmission process.

The findings we discovered have implications for future works as well. First, they show the balance between data accuracy and speed, especially when dealing with large amounts of information which becomes more crucial as autonomous systems become more complex. Secondly, our results provide insights for designing networks in ROS 2 emphasizing the importance of effectively managing flow control to meet different performance needs.

Moreover, it's essential to consider the roles of TCP and UDP protocols in ROS 2 systems. Unlike ROS 1 that depended on a ROS master for discovery the `large_data` feature in ROS 2 maintains an automatic and decentralized discovery process while benefiting from TCP's reliability without sacrificing server endpoint transparency.

However, this feature isn't activated by default because of reasons we will discuss. One key consideration is ensuring compatibility with DDS based RMWs—an essential aspect of ROS 2s adaptability. Also UDPs real-time responsiveness remains critical for a range of applications within and beyond robotics. The capability to send data samples over potentially unreliable networks without excessive overhead marks a significant advancement, for ROS 2. However the effectiveness of this function depends on the application it is used for. Choosing between TCP and UDP should be based on the needs and goals of each project.

As a result, while this study shows that TCP improves the reliability of transmitting amounts of data it also highlights the importance of UDP in situations where real-time data sharing is crucial. The discussion in the ROS community should acknowledge the roles played by both protocols allowing for a more adaptable and resilient network structure, within ROS 2 environments.

4.2 Future Works

As our analysis has laid the groundwork for understanding ROS 2 system performance, the next logical step is to integrate real-time data from autonomous aerial and ground vehicles. This integration will not only test the effectiveness of our established policies but also edge us closer to realizing a secure DDS network capable of simulating a variety of threat scenarios, including DDoS and other forms of cyber-attacks. The resilience and responsiveness of ROS fleets to such adversities remain a critical area for future research.

Additionally, we must explore and create the blend use of UDP and TCP protocols within ROS 2 to optimize network performance. Envision a hybrid network where UDP's immediacy benefits local domain communications, while TCP ensures the integrity of extensive data transmitted across multiple domains, preserving every critical frame from loss.

Presently, the DDS specification doesn't allow different QoS policies for separate topics within ROS, a limitation that constrains the adaptability of data transfer. This represents a significant opportunity for enhancement. By developing a system that supports dual QoS policies tailored to the specific needs of each data topic, we can significantly enhance the DDS layer's flexibility. Such an advancement would permit simultaneous execution of real-time operations and reliable large-scale data transmission, heralding a new era of robustness and versatility in multi-hop DDS networks.

Appendices

Appendix A ROS 2 custom talker with Custom QoS.

```
#include <chrono>
#include <cstdio>
#include <memory>
#include <sstream>
#include <string>
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

class CustomTalker : public rclcpp::Node {
public:
    CustomTalker(const std::string & topic_name, int payload_size_kb)
    : Node("custom_talker"), topic_name_(topic_name), payload_size_kb_(payload_size_kb), total_bytes
        // Define custom QoS profile for the publisher
        auto custom_qos = rclcpp::QoS(rclcpp::KeepLast(10)).best_effort().durability_volatile().dead

        // Initialize the publisher with custom QoS
        publisher_ = this->create_publisher<std_msgs::msg::String>(topic_name_, custom_qos);

        timer_ = this->create_wall_timer(
            std::chrono::milliseconds(500),
            [this]() { publish_message(); });
    }

    ~CustomTalker() {
        auto end_time = std::chrono::steady_clock::now();
        std::chrono::duration<double> elapsed_seconds = end_time - start_time_;
        double throughput = total_bytes_sent_ / elapsed_seconds.count();
        RCLCPP_INFO(this->get_logger(), "Total Messages Published: %d, Total Bytes Sent: %ld, Elapse
                    i_, total_bytes_sent_, elapsed_seconds.count(), throughput);
```

```
    }

private:
    void publish_message() {
        auto now = std::chrono::system_clock::now();
        auto timestamp = std::chrono::duration_cast<std::chrono::milliseconds>(now.time_since_epoch(

        int payload_size_bytes = payload_size_kb_ * 1024;
        total_bytes_sent_ += payload_size_bytes;

        std::string payload(payload_size_bytes, 'a');
        std_msgs::msg::String message;
        message.data = "Seq: " + std::to_string(i_) + ", Time: " + std::to_string(timestamp) + ", "

        publisher_->publish(message);

        RCLCPP_INFO(this->get_logger(), "%s Publishing sequence %d with timestamp %ld and payload si
                    topic_name_.c_str(), i_, timestamp, payload_size_bytes);

        i_++;
    }

    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
    rclcpp::TimerBase::SharedPtr timer_;
    std::string topic_name_;
    int payload_size_kb_;
    long total_bytes_sent_;
    std::chrono::steady_clock::time_point start_time_;
    int i_;
};
```

```
int main(int argc, char * argv[]) {
    rclcpp::init(argc, argv);

    if (argc < 3) {
        std::cerr << "Usage: ./custom_talker <topic_name> <payload_size_kb>" << std::endl;
        return 1;
    }

    std::string topic_name = argv[1];
    int payload_size_kb = std::stoi(argv[2]);

    auto node = std::make_shared<CustomTalker>(topic_name, payload_size_kb);
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

Appendix B ROS 2 Repeater Code with Custom QoS

```
#include <chrono>
#include <memory>
#include <string>
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

class RepeaterNode : public rclcpp::Node {
public:
    RepeaterNode(const std::string& subscription_topic, const std::string& publication_topic)
    : Node("repeater_node"), total_bytes_processed_(0), start_time_(std::chrono::steady_clock::now()
        // Define custom QoS profile for both subscription and publisher with BEST_EFFORT reliabilit
        auto custom_qos = rclcpp::QoS(rclcpp::KeepLast(10)).best_effort().durability_volatile().dead

        // Subscribe with custom QoS
        subscription_ = this->create_subscription<std_msgs::msg::String>(
            subscription_topic,
            custom_qos,
            [this, subscription_topic](const std_msgs::msg::String::SharedPtr msg) {
                auto receive_time = std::chrono::steady_clock::now();
                size_t msg_size = msg->data.length();
                total_bytes_processed_ += msg_size;

                // Repeat (publish) the message with the same QoS
                publisher_->publish(*msg);

                auto publish_time = std::chrono::steady_clock::now();
                std::chrono::duration<double, std::milli> processing_time = publish_time - receive_t
                total_processing_time_ms_ += processing_time.count();
                message_count_++;
```

```
                // Log for each message; consider removing or modifying for high throughput scenario
                RCLCPP_INFO(this->get_logger(), "Received %zu bytes from %s and processed in %f ms",
                            msg_size, subscription_topic.c_str(), processing_time.count());
            });

        // Initialize the publisher with custom QoS
        publisher_ = this->create_publisher<std_msgs::msg::String>(publication_topic, custom_qos);
    }

    ~RepeaterNode() {
        auto end_time = std::chrono::steady_clock::now();
        std::chrono::duration<double> elapsed_seconds = end_time - start_time_;
        double throughput = total_bytes_processed_ / elapsed_seconds.count(); // Bytes per second
        double average_processing_time_ms = message_count_ > 0 ? total_processing_time_ms_ / message

        RCLCPP_INFO(this->get_logger(), "Total Bytes Processed: %ld, Elapsed Time: %f seconds, Throu
                    total_bytes_processed_, elapsed_seconds.count(), throughput, average_processing_
    }

private:
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
    long total_bytes_processed_;
    std::chrono::steady_clock::time_point start_time_;
    double total_processing_time_ms_; // Total processing time in milliseconds
    long message_count_; // Number of messages processed
};

int main(int argc, char * argv[]) {
    rclcpp::init(argc, argv);
```

```
    if (argc < 3) {
        RCLCPP_ERROR(rclcpp::get_logger("rclcpp"), "Usage: repeater <subscription_topic> <publicatio
        return 1;
    }

    auto node = std::make_shared<RepeaterNode>(argv[1], argv[2]);
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

Appendix C ROS 2 custom listener with Custom QoS.

```
#include <chrono>
#include <iostream>
#include <memory>
#include <sstream>
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

class CustomListener : public rclcpp::Node {
public:
    CustomListener(const std::string& topic_name)
    : Node("custom_listener"),
      topic_name_(topic_name),
      total_bytes_received_(0),
      total_delay_ms_(0),
      message_count_(0),
      start_time_(std::chrono::steady_clock::now()) {
        // Define custom QoS profile for the subscription
        auto custom_qos = rclcpp::QoS(rclcpp::KeepLast(10)).best_effort().durability_volatile().dead

        // Subscribe with custom QoS
        subscription_ = this->create_subscription<std_msgs::msg::String>(
            topic_name_,
            custom_qos,
            [this](const std_msgs::msg::String::SharedPtr msg) {
                auto receipt_time = std::chrono::system_clock::now();
                auto receipt_timestamp = std::chrono::duration_cast<std::chrono::milliseconds>(
                    receipt_time.time_since_epoch()).count();

                std::istringstream message_stream(msg->data);
```

```
                std::string part;
                long sent_sequence_num = 0;
                std::getline(message_stream, part, ','); // Extract sequence number
                sscanf(part.c_str(), "Seq: %ld", &sent_sequence_num);
                long sent_timestamp = 0;
                std::getline(message_stream, part, ','); // Extract sent timestamp
                sscanf(part.c_str(), " Time: %ld", &sent_timestamp);

                long delay_ms = receipt_timestamp - sent_timestamp;
                total_delay_ms_ += delay_ms; // Accumulate delay
                message_count_++; // Increment message count
                total_bytes_received_ += msg->data.length(); // Update total bytes received

                RCLCPP_INFO(this->get_logger(), "Listening: Received message with Seq. Num %ld, Dela
                            sent_sequence_num, delay_ms, total_bytes_received_);
            });
    }

    ~CustomListener() {
        auto end_time = std::chrono::steady_clock::now();
        std::chrono::duration<double> elapsed_seconds = end_time - start_time_;
        double average_delay_ms = message_count_ > 0 ? static_cast<double>(total_delay_ms_) / messag
        double throughput = total_bytes_received_ / elapsed_seconds.count(); // Bytes per second

        RCLCPP_INFO(this->get_logger(), "Session ended. Total Messages Received: %d, Average Delay:
                    message_count_, average_delay_ms, total_bytes_received_, throughput);
    }

private:
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
    std::string topic_name_;
```

```
    std::chrono::steady_clock::time_point start_time_;
    long total_bytes_received_;
    long total_delay_ms_; // Accumulated total delay in milliseconds
    int message_count_; // Number of messages received
};

int main(int argc, char * argv[]) {
    rclcpp::init(argc, argv);

    if (argc < 2) {
        std::cerr << "Usage: custom_listener <topic_name>" << std::endl;
        return 1;
    }

    std::string topic_name = argv[1];
    auto node = std::make_shared<CustomListener>(topic_name);
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

Bibliography

[1] Recommendation x.500. Technical report, International Telecommunication Union, Nov 1988.

[2] Recommendation x.509. Technical report, International Telecommunication Union, Nov 1988.

[3] Websites using ssl by default, Dec 2023. URL or organization details if available.

[4] About Quality of Service Settings. `https://docs.ros.org/en/iron/Concepts/Intermediate/About-Quality-of-Service-Settings.html`, 2024. Accessed: 2024-03-31.

[5] D. Axe. Ukrainian marines hacked a russian drone to locate its base—then blew up the base with artillery, Nov 2023. Accessed: insert date of access.

[6] Ivan Baldin, Andy Nikolich, Jim Griffioen, Inder Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. FABRIC: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing*, 23(6):38–47, 2019.

[7] E. Barker. *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms*. National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, Mar 2020. Federal Information Processing Standards Publication 46.

[8] S. Berkovits, S. Chokhani, J. Furlong, J. Geiter, and J. Guild. Public key infrastructure (final report). Technical report, National Institute of Standards and Technology, Gaithersburg, MD, Apr 1994. Federal Information Processing.

[9] J. Daemen and V. Rijmen. Aes proposal: Rijndael. Technical report, 1999.

[10] Department of Defense. Unmanned systems integrated roadmap: 2017–2042. `https://www.defensedaily.com/wp-content/uploads/post_attachment/206477.pdf`, n.d. Accessed: insert date of access.

[11] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[12] V. Diluoffo, W.R. Michalson, and B. Sunar. Robot operating system 2: The need for a holistic security approach to robotic architectures. *International Journal of Advanced Robotic Systems*, 15(3):1–15, 2018.

[13] eProsima. Handling large data with fast dds. `https://fast-dds.docs.eprosima.com/en/latest/fastdds/use_cases/large_data/large_data.html`, 2021. Accessed: 2023-03-31.

[14] eProsima. Tcp transport use case - large data. `https://fast-dds.docs.eprosima.com/en/latest/fastdds/use_cases/tcp/tcp_large_data_with_options.html`, 2021. Accessed: 2023-04-01.

[15] eProsima. Large data. https://fast-dds.docs.eprosima.com/en/latest/fastdds/use_cases/large_data/large_data.html#flow-controllers, 2023. Accessed: 2023-09-27.

[16] eProsima. Fast dds discovery documentation. https://fast-dds.docs.eprosima.com/en/latest/fastdds/discovery/discovery.html, 2024. Accessed: 2024-3-15.

[17] eProsima Fast DDS. Ros 2 integration. https://fast-dds.docs.eprosima.com/en/latest/fastdds/ros2/ros2.html, 2024. Accessed: 2024-02-28.

[18] Horst Feistel. Cryptographic coding for data-bank privacy. Technical Report RC2827, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Mar 1970.

[19] J. Fernandez, B. Allen, P. Thulasiraman, and B. Bingham. Performance study of the robot operating system 2 with qos and cyber security settings. In *2020 IEEE International Systems Conference (SysCon)*, pages 1–6, 2020.

[20] W. Ford. Public-key infrastructure interoperation. In *1998 IEEE Aerospace Conference Proceedings*, volume 4, pages 329–333, 1998.

[21] Kyu haeng Lee, Chong kwon Kim, Kyeong Tae Kim, and Won tae Kim. Router design for dds: Architecture and performance evaluation. In *2014 International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 250–254, 2014.

[22] E. Hansen. Analysis of design for implementing confidentiality, integrity, authentication, and non-repudiation solutions. SANS Institute, Jun 2003.

[23] Martin E Hellman and Whitfield Diffie. Special feature exhaustive cryptanalysis of the nbs data encryption standard. *Computer*, 10(6):74–84, Jun 1977.

[24] L. M. Kohnfelder. Towards a practical public-key cryptosystem. Master's thesis, MIT, May 1978.

[25] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th International Conference on Embedded Software*, EMSOFT '16, New York, NY, USA, 2016. Association for Computing Machinery.

[26] Object Management Group (OMG). Dds security, version 1.1. Technical report, Object Management Group, April 2018. OMG Document Number: formal/2018-04-01.

[27] National Bureau of Standards. Data encryption standard. Technical Report FIPS PUB 46, National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161, Jan 1977.

[28] Open Robotics. Intrinsic acquires osrc and osrc sg. https://www.openrobotics.org/blog/2022/12/15/intrinsic-acquires-osrc-and-osrc-sg, 12 2022. Accessed: 2024-02-28.

[29] G. Pardo and R. White. Leveraging dds security in ros2. In *Presented at ROSCon 2018*, 2018.

[30] Real-Time Innovations. *Large Data and Fragmentation*. Real-Time Innovations, 7.2.0 edition, February 2022. Accessed: 2023-03-31.

[31] ROS Documentation. About logging. https://docs.ros.org/en/rolling/Concepts/Intermediate/About-Logging.html, 2022. Accessed: 2024-02-28.

[32] ROS Industrial. Ros industrial. https://wiki.ros.org/Industrial, 2024. Accessed: 2024-02-28.

[33] ROS Military. Ros military. https://rosmilitary.org/, 2024. Accessed: 2024-02-28.

[34] RTI. Who is chopping my application data and why should i care?, July 2017. Accessed: [Insert access date here].

[35] Paul Ruth, Ivan Baldin, Koustubh Thareja, Tom Lehman, X. Yang, and Ezra Kissel. Fabric network service model. In *2022 IFIP Networking Conference (IFIP Networking)*, pages 1–6, 2022.

[36] S. Sandoval and P. Thulasiraman. Cyber security assessment of the robot operating system 2 for aerial networks. In *Proc. of IEEE International Systems Conference (SYSCON)*, 2019.

[37] J. L. Smith. The design of lucifer, a cryptographic device for data communications. Technical Report RC3326, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Apr 1971.

[38] A. Sorkin. Lucifer, a cryptographic algorithm. *Cryptologia*, 8(1):22–42, 1984.

[39] Kai Weng Wong and Hadas Kress-Gazit. From high-level task specification to robot operating system (ros) implementation. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 188–195, 2017.