2020

Performance Study of the Robot Operating System 2 with QoS and Cyber Security Settings

Fernandez, J.; Allen, B.; Thulasiraman, P.; Bingham, B.

IEEE

Fernandez, J., et al. "Performance Study of the Robot Operating System 2 with QoS and Cyber Security Settings." 2020 IEEE International Systems Conference (SysCon). IEEE, 2020.

http://hdl.handle.net/10945/69143

Performance Study of the Robot Operating System 2 with QoS and Cyber Security Settings

J. Fernandez, B. Allen, P. Thulasiraman and B. Bingham
Naval Postgraduate School, Monterey, CA, USA
jmfernan1@nps.edu, bdallen@nps.edu, pthulas1@nps.edu, bbingham@nps.edu

***Abstract*— Throughout the Department of Defense, there are ongoing efforts to increase cybersecurity and improve data transfer in unmanned robotic systems (UxS). This paper explores the performance of the Robot Operating System (ROS) 2, which is built with the Data Distribution Service (DDS) standard as a middleware. Based on how quality of service (QoS) parameters are defined in the robotic middleware interface, it is possible to implement strict delivery requirements to different nodes on a dynamic nodal network with multiple unmanned systems connected. Through this research, different scenarios with varying QoS settings were implemented and compared to baseline values to help illustrate the impact of latency and throughput on data flow. DDS security settings were also enabled to help understand the cost of overhead and performance when secured data is compared to plaintext baseline values. Our experiments were performed using a basic ROS 2 network consisting of two nodes (one publisher and one subscriber). Our experiments showed a measurable latency and throughput change between different QoS profiles and security settings. We analyze the trends and tradeoffs associated with varying QoS and security settings. This paper provides performance data points that can be used to help future researchers and developers make informative choices when using ROS 2 for UxS.**

***Keywords*-unmanned systems, security, robotic systems, ROS**

I. Introduction

The increased use of unmanned systems (UxS) in warfare centric environments makes it increasingly vulnerable to cyber threats. In 2017, the Office of the Secretary Defense (OSD) released their UxS Roadmap in which Data Transport Integration and Cyber Security were identified as two key challenges for UxS in the upcoming decades [1]. In order to reduce the costs associated with new software development for data migration, the Defense Science Board has pushed the adoption of open architectures (OA) in UxS development [2].

One such architecture is the Robot Operating System (ROS), which includes an open source middleware system for robots. The first version of ROS, ROS 1, was developed by Open Source Robotics Foundation (OSRF) in 2008. ROS 1 is a valuable platform to conduct research, but the architecture does not provide any native security between nodes, which is crucial for mission support and tactical deployment.The second version of ROS, ROS 2, introduces a security architecture that provides the user the ability to enable and use security features in order to cyber harden a system. This is imperative within today's DoD Information Assurance (IA) requirements [1]. However, one must understand the costs (latency, throughput, overhead, etc.) associated with these IA requirements. The tradeoff between system security and system performance must be addressed to ensure timely and effective execution of operational tasks [3].

A. Research Motivations and Contributions

The objective of this work is to quantify the performance of ROS 2 operating in a two node network while applying different quality of service (QoS) profiles and security settings. We provide an in-depth study on the different QoS profiles available to ROS 2 in the context of network performance. We also investigate the impact of the ROS 2 security plugins on network performance. ROS 2 is built on top of the Data Distribution Service (DDS) as its middleware. The DDS is a well understood, industry standard that has been implemented by a number of vendors and deployed in numerous industrial and defense applications. A key aspect of ROS 2 is the ROS middleware (RMW) interface between the ROS 2 software stack and vendor-specific DDS implementations. This integration of ROS 2 with DDS has not been extensively studied from the perspective of network/system performance.

To the best of our knowledge, this is the first paper to comprehensively study the performance of ROS 2 in terms of both service delivery and security. The performance results obtained when different QoS and security settings are enabled will contribute towards evaluating and configuring ROS 2 in UxS for different use cases.

The contributions of this paper are:

- Analysis and experimentation of ROS 2 performance using varying QoS profile combinations for plaintext data traffic. Performance is measured in terms of 1) packet loss; 2) latency; 3) throughput; and 4) overhead generation.
- Analysis and experimentation of ROS 2 performance using varying QoS profile combinations for encrypted data traffic. Performance is measured in terms of 1) packet loss; 2) latency; 3) throughput; and 4) overhead generation.

This paper does not provide specific recommendations of how to utilize ROS 2 for specific use cases, but instead provides a series of performance measurements with different QoS and security combinations that developers can use to optimize ROS 2 performance.

The rest of this paper is organized in the following manner. In Section II, we discuss the relevant background information

978-1-7281-5365-0/20/$31.00 ©2020 IEEE

on ROS 2 network performance in terms of QoS and security. In Section III we describe our experimental setup, including design, implementation, and execution. In Section IV, we present our results and analysis from multiple experimental runs. In Section V, we conclude the paper.

II. Related Work

ROS 2 provides a common robotic software infrastructure that mitigates the time for code development and interoperability issues. One of the key features of ROS 2 with DDS is the use of the Real-Time-Publish Subscribe (RTPS) communication standard. RTPS provides configurability and scalability in a real time data environment. These features translate to improved latency and throughput as seen in the eProsima Fast RTPS performance tests [4] (eProsima^{TM}is a vendor of DDS middleware). The DDS RTPS protocol can also be configured with different QoS and security settings.

A. ROS 2 Quality of Service

The recent ROS 2 Crystal Clemmys release (used in this paper) natively supports three different QoS policies. For eProsima, these QoS policies are:

- Reliability: Two different sub-policies fall under the umbrella of reliability. 1) BEST_EFFORT: messages are sent without arrival confirmation from the receiver. This has the fastest delivery but messages can be lost; 2) RELIABLE: the publisher expects arrival confirmation from the receiver. This is a slower method that prevents data loss [5].
- History: This policy refers to message caching. There are two sub-policies for sample/data storage. 1) KEEP_ALL: stores all samples/data in memory; 2) KEEP_LAST: stores samples/data up to a maximum queue depth. Queue depth is a configurable option in DDS [5].
- Durability: This policy defines how a node behaves regarding samples/data that existed on a topic before the subscriber joined. Three sub-policies exist. 1) VOLATILE: past samples/data are ignored and the subscriber receives samples/data after the moment it joins; 2) TRANSIENT_LOCAL: when a new subscriber joins, its History (queue) is filled with past samples/data that were stored locally; 3) TRANSIENT: when a new subscriber joins its History (queue) is filled with past samples/data which are stored in persistent storage.

In [6], the authors measured end-to-end latencies and throughput on ROS 1 and ROS 2. Only two QoS profiles were used for all ROS 2 experiments.The authors showed that latency differed greatly as data size increased but remained similar at small data values [6].

B. ROS 2 Security

In [7], the authors conducted a review of using DDS with ROS 2. The group looked at data sent as plaintext versus full security enabled data using Rivest-Shamir-Adleman (RSA) 2048 bit and Elliptic Curve Cryptography (ECC) 256 bit. It was found that, regardless of algorithm or key size, the overhead of security enabled data had an average increase of approximately 137% in latency performance and 132% in the number of packets transmitted.

At the 2018 ROSCon conference, researchers presented the performance impact due to enabling security in DDS, specifically by vendor Real Time Innovations (RTI) [8]. The data size was also increased. The authors showed that latency increases as more security features (i.e., signing, hashing, encryption etc.) are added [8].

In [9], we focused on the viablity of ROS 2 to safeguard communications between multiple unmanned aerial vehicles and a ground control station (GCS). We specifically looked at the three security plugins to verify ROS 2's ability to mitigate multiple threats. The work in [9] was not investigated from a network performance perspective. The emphasis of [9] was on the verification of ROS 2's security elements on a small unmanned aerial system.

This paper builds upon the work of [9] by focusing on the DDS security architecture and service delivery profiles available to ROS 2. While all of the studies in the literature have looked at either QoS or security individually, none have investigated the combination of QoS and security together on ROS 2 network performance.

III. System Design and Setup

All experiments were executed using a single topic and participant setup consisting of one publisher and one subscriber node as illustrated in Fig. 1. The default installed RMW implementation, eProsima Fast RTPS, was used in our experiments.

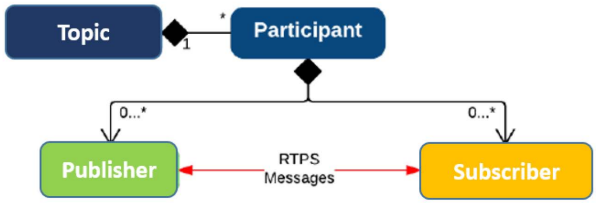

Fig. 1. System setup with RTPS. Source [4]

Experiments in this paper were performed on a SYSTEM76 Wild Dog Pro desktop with a 4.6 GHz i7-8700 processor running Ubuntu 18.04 LTS (Bionic Beaver) with ROS 2 binaries Crystal Clemmys patch release 2 (February 2019). Wireshark version 2.6.6 was used to capture and analyze all one-way network traffic on the loopback internet protocol (IP) address 127.0.0.1.

A. Quality of Service Settings

Our experiments tested five different QoS profiles for plaintext and secure data. The parameters of each profile are defined in Table I. These parameters were applied to both node participants in order to avoid compatibility issues between the two participants.

The first two cases both use BEST_EFFORT reliability but with differing history and durability settings. These cases were

TABLE I

QoS PROFILES SUMMARY FOR THE PUBLISHER/SUBSCRIBER NODES

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

designed to determine the impact of the history parameter on performance metrics. For Case (a), depth was set to five and durability was set to VOLATILE. This means that the amount of data saved in the history cache is set to five packets and the VOLATILE setting requires that no old data be sent to a new subscriber participant that joins in the middle of transmissions.

For Case (b), both participants attempt to maintain a complete history in the cache (KEEP_ALL). The TRANSIENT_LOCAL setting was selected as this allows a subscriber to join the topic late and receive previously sent messages up to the limits set in depth. This parameter setting takes more resources to implement resulting in possible lower performance. In summary, Cases (a) and (b) are set to opposite ends of the performance spectrum (Case (a) results in fast data transmission while Case (b) results in slow data transmissions) under BEST_EFFORT reliability.

Cases (c), (d), and (e) were setup in a similar manner to Case (a) and (b) in terms of anticipated performance. For these three cases, the reliability parameter was set to RELIABLE. In these cases, heartbeat (HB) messages and acknowledgements (ACKNACKs) are sent from the publisher and subscriber throughout the session, and any unacknowledged data samples will result in retransmissions from the subscriber. The HB is a submessage sent from the publisher to the subscriber that checks for connectivity between the two nodes. The ACKNACK notifies the publisher of which packet sequence numbers the subscriber has received and which packet sequence numbers remain missing [10]. It is still possible to have lost data samples if the history depth is not large enough to allow for re-transmissions.

B. Security Settings

eProsima defines the three security plugins as follows:

- Authentication: Provides authentication between discovered participants. Authentication is achieved with a trusted Certificate Authority (CA) and implements Elliptic Curve Digital Signature Algorithm (ECDSA) to perform the mutual authentication. It also establishes a shared secret key using Elliptic Curve Diffie-Hellman (ECDH) [4].
- Access Control: Provides validation of entity permissions and access control using a permissions document signed by a shared CA [4].
- Cryptographic: Provides encryption applied over the whole RTPS messages. The default ECC 256 encryption algorithm in ROS 2 was used for our experiments. OpenSSL software library and commands were used to generate the ECC certificates and public and private keys for both the publisher and subscriber. By default, in this paper, no access restrictions were set.

IV. RESULTS

One thousand messages of varying sizes were sent in both plain and encrypted text format between the publisher and subscriber. Latency was measured using the transmission time plus the time the middleware takes to process a message, encrypt and serialize the data for transmission. All latency and throughput values were compared against Case 1(a) values (Table I) in a percentage format per Equation (1) and (2). Case 1(a) was chosen as the baseline since it was expected to have the best performance for all message sizes in plaintext data.

$$latency\Delta\% = \frac{latency_{new} - latency_{1a}}{latency_{1a}} \times 100\% \quad (1)$$

$$throughput\Delta\% = \frac{throughput_{new} - throughput_{1a}}{throughput_{1a}} \times 100\% \quad (2)$$

While modifying the QoS and security settings, overhead values due to excess protocol data generation, and packet retransmission can be observed. We define overhead as the number of data packets that are sent in addition to actual message fragments. These excess data packets primarily consist of metadata that is not appended to RTPS message fragments (metadata consists of HBs and ACKNACKs).

A. Simulation Parameter Definitions

The following provides definitions for each column of our tabulated results.

- Total Packets: Packets counted in Wireshark from the first transmitted RTPS message fragment until the last transmitted message fragment. Includes all metadata (HBs and ACKNACKs) and discovery protocol messages transmissions after fragment one.
- Message (MSG) Fragment Packets: Packets are fixed size. Packets are counted in Wireshark that only include RTPS message fragments.
- Overhead Packets (%): The ratio of overhead packet messages divided by the total packets.
- MSGs Lost: Number of messages that were not received by the subscribing node. This could be due to a lost fragment, a lost message, data collision, etc.
- MSG Fragment Latency (μs): The latency for each transmitted RTPS fragment was calculated by subtracting the timestamp from the current message fragment from the previously transmitted message fragment.
- MSG Latency (μs): RTPS message fragments were added up to determine the total latency for transmitting one message. This value was then averaged across the other 999 messages transmitted, including some retransmitted fragments and other messages with incomplete fragments.

TABLE II

0.25 MB OVERHEAD RESULTS

Case	File Size (MB)	Total Packets	MSG Frag Packets	Overhead Packets (%)	MSGs Lost
1a	0.25	3672	3627	1.23	2
1b		4048	3999	1.21	12
1c		5989	3979	33.56	4
1d		5023	3949	21.38	0
1e		4843	3782	21.91	0
2a	0.25	3561	3512	1.38	122
2b		3847	3800	1.21	50
2c		3705	3601	2.81	103
2d		3646	3537	2.99	121
2e		3621	3516	2.90	129

- MSG Throughput (Gbps): The throughput calculation is given as $Throughput(Gbps) = \frac{MSGsize(bits)}{latency(\mu s)}$. The size of the message is equal to the total size of the message fragments, added together.
- $\Delta\%$: Each $\Delta\%$ is a comparison for either the average MSG fragment latency, average MSG latency, or average throughput compared against Case 1(a) by using Equations (1) or (2).

B. Simulation Results

1) 0.25MB File Size: Table II displays the overhead data for plaintext (Cases 1(a)-1(e)) and secure data (Cases 2(a)-2(e)) as well as messages lost during transmission. Table III displays all latencies and throughputs for the ten 0.25MB experimental runs.

In Table II, for Cases 1(a) and 1(b), little overhead was generated when compared to Cases 1(c), 1(d), and 1(e). This was because metadata was not appended to the RTPS message fragments (as was in Cases 1(c), 1(d), and 1(e)), but was transmitted separately. For the RELIABLE Cases (c, d, and e), it was expected that zero messages would be lost as long as a sufficient history depth is set. For Case 1(d) and 1(e), we see zero messages lost. Case 1(c) has some messages lost, as expected, since depth is set to five. For the secure cases (Cases 2(a)-2(e)), there is an unexpected high number of lost messages. For this file size, message losses for the secure cases were much higher than that of the plaintext cases. The tests for the secure RELIABLE cases were run multiple times to ensure the results were accurate. The number of lost messages stayed the same during each experimental trial (greater than 10%). We assumed that given the RELIABLE QoS setting, messages would be re-transmitted to ensure delivery. However, these lost messages indicate that the re-transmissions timed out and could not be delivered.

Table III displays performance metrics related to latency and throughput. Case 1(a) has no data displayed in the $\Delta\%$ columns since this case was considered the baseline for comparison. Case 1(b) and 2(b) have the worst performance metrics when compared to the baseline. This is when QoS settings were set to BEST_EFFORT, KEEP_ALL, and TRANSIENT_LOCAL.

TABLE III

0.25 MB LATENCY AND THROUGHPUT RESULTS

Case	File Size (MB)	MSG Frag Latency (µs)	Δ%	MSG Latency (µs)	Δ%	MSG Throughput (Gbps)	Δ%
1a	0.25	25.8	-	105.0	-	2.477	-
1b		40.4	56.6	162	54.3	1.590	-35.8
1c		29.5	14.3	118	12.4	2.130	-14.0
1d		32.3	25.2	129.6	23.4	1.979	-20.1
1e		30.3	17.4	128.1	22.0	2.127	-14.1
2a	0.25	62.7	143.0	250.7	138.8	0.988	-60.1
2b		84.9	229.1	345.6	229.1	0.730	-70.5
2c		70.7	174.0	282.2	168.8	0.876	-64.6
2d		74.7	189.5	297.7	183.5	0.850	-65.7
2e		72.3	180.2	287.9	174.2	0.861	-65.2

TABLE IV

0.5 MB OVERHEAD RESULTS

Case	File Size (MB)	Total Packets	MSG Frag Packets	Overhead Packets (%)	MSGs Lost
1a	0.50	7094	7046	0.68	7
1b		8040	7994	0.57	19
1c		13531	7831	42.13	3
1d		10633	7614	28.39	0
1e		11119	8004	28.02	0
2a	0.50	7609	7560	0.64	55
2b		7851	7807	0.56	25
2c		16828	16521	1.82	49
2d		7846	7682	2.09	58
2e		7814	7650	2.10	39

2) 0.5MB File Size: In Table IV, the trend of the overhead results are very similar to the 0.25MB results. Case 1(d) and 1(e) continue to have 100% message delivery (0 messages lost) but Case 2(d) and Case 2(e) continue to have loses. As the message size approximately doubled in size, the message losses for all secured transmitted data have approximately decreased by the half (see last column of Table II vs Table IV).

Table V expressed very similar trends from Table III, including Case 1(b) and 2(b) continuing to display the worst performance metrics compared to Case 1(a). Throughput stayed nearly constant as history depth increased from five to 1000 in the RELIABLE cases (Cases 1(c) vs 1(d) and Cases 2(c) vs 2(d)). However, latency increases significantly for these cases.

3) 1MB File Size: In Table VI, Case 1(c) continues to have a large amount of retransmitted metadata (overhead is 43.89%)

TABLE V

0.5 MB LATENCY AND THROUGHPUT RESULTS

Case	File Size (MB)	MSG Frag Latency (µs)	Δ%	MSG Latency (µs)	Δ%	MSG Throughput (Gbps)	Δ%
1a	0.50	26.3	-	242.6	-	2.459	-
1b		40.8	55.1	335	38.1	1.554	-36.8
1c		30.4	15.6	242	0.2	2.085	-15.2
1d		32.2	22.4	260.5	7.4	2.024	-17.6
1e		30.3	15.2	243.4	0.3	2.125	-13.6
2a	0.50	65.8	150.2	522.2	115.3	0.959	-61.0
2b		90.5	244.1	730.8	201.2	0.700	-71.5
2c		72.8	157.2	1233	177.8	0.876	-61.3
2d		71.1	170.3	563.1	132.1	0.889	-63.8
2e		71.5	171.9	577.8	138.2	0.884	-64.1

TABLE VI

1MB OVERHEAD RESULTS

Case	File Size (MB)	Total Packets	MSG Frag Packets	Overhead Packets (%)	MSGs Lost
1a	1	15527	15487	0.26	10
1b		15393	15357	0.23	17
1c		27716	15552	43.89	28
1d		23315	16077	31.04	0
1e		23892	16491	30.98	0
2a	1	15871	15824	0.30	41
2b		15787	15744	0.27	19
2c		16828	16521	1.82	49
2d		16724	16419	2.82	16
2e		16740	16306	2.59	19

TABLE VII

1MB THROUGHPUT AND LATENCY RESULTS

Case	File Size (MB)	MSG Frag Latency (µs)	Δ%	MSG Latency (µs)	Δ%	MSG Throughput (Gbps)	Δ%
1a	1	28.3	-	443.8	-	2.261	-
1b		45.2	59.7	722.2	62.7	1.451	-35.8
1c		33.2	17.3	529.9	19.4	1.938	-14.3
1d		36.1	27.6	608.7	37.2	1.869	-17.3
1e		33.1	17.0	558.5	25.8	1.986	-12.2
2a	1	68.3	141.3	1065	140.0	0.935	-58.6
2b		84.2	197.5	1347	203.5	0.759	-66.4
2c		72.8	157.2	1233	177.8	0.876	-61.3
2d		75.1	165.4	1249	181.4	0.864	-61.8
2e		74.0	161.5	1240	179.4	0.868	-61.6

TABLE VIII

2MB OVERHEAD RESULTS

Case	File Size (MB)	Total Packets	MSG Frag Packets	Overhead Packets (%)	MSGs Lost
1a	2	30127	30089	0.13	77
1b		29024	28986	0.13	24
1c		55504	32062	42.23	99
1d		50167	33817	32.59	0
1e		49952	33700	32.54	0
2a	2	31867	31824	0.13	17
2b		32043	32000	0.13	42
2c		33908	33332	1.70	22
2d		56632	56228	0.71	0
2e		43930	43383	1.25	0

TABLE IX

2MB THROUGHPUT AND LATENCY RESULTS

Case	File Size (MB)	MSG Frag Latency (µs)	Δ%	MSG Latency (µs)	Δ%	MSG Throughput (Gbps)	Δ%
1a	2	34.8	-	1029	-	1.868	-
1b		48.6	39.7	1471	43.0	1.369	-26.7
1c		37.4	7.5	1266	23.0	1.767	-5.4
1d		38.1	9.5	1317	28.0	1.720	-7.9
1e		36.9	6.0	1313	27.6	1.806	-3.3
2a	2	75.2	116.1	2139	107.9	0.846	-54.7
2b		90.5	244.1	730.8	201.2	0.700	-71.5
2c		77.8	123.6	2591	151.8	0.830	-55.5
2d		81.3	133.6	3497	239.8	0.751	-59.8
2e		79.7	129.0	3147	205.8	0.799	-57.2

as was seen in previous file sizes. This was expected due to the RELIABLE setting and history depth set to five. The subscriber continually informs the publisher of the missing data, but since history depth is so small, the publisher is unable to retransmit from the temporary cache. Overall, for the secure data, the message losses are continuing to decrease as the message size increases. For Case 2(d) and 2(e), as the message size doubles, the messages lost decreases significantly when compared to Tables II and IV. This follows the trend that was seen for the 0.5MB file size. All other trends appear relatively the same.

From Table VII, the RELIABLE cases maintain performance results for fragment latency and overall throughput with metrics appearing to start converging towards each other as file size increases.

4) 2MB File Size: In Table VIII, the overhead metrics maintain the same trend as seen in previous messages sizes, but now message losses are increasing by a significant amount compared to the 1MB file size. For the secure data cases, this is the first time all messages were delivered for the Case 2(d) and Case 2(e) data runs (0 messages lost). Experiments for these two cases were run multiple times to ensure these were accurate results. The tradeoff of achieving 100% message delivery was the very high increase in message fragment retransmissions required to achieve this. We manually adjusted the file size to determine when 100% reliability could be achieved for this experimental setup. At an approximate 1.25MB file size, a continuous 100% message delivery is achieved for Case 2(d) and 2(e). What this shows is that secure data is more effective when the message size is larger. This is an interesting result that requires further study into the dynamics of the ROS 2 security plugins that is causing re-transmissions to fail at smaller data sizes. In Table IX, previously discussed latency and throughput trends remain with this message size.

5) 4MB File Size: Due to space, we do not show the tabulated results for the 4MB file size. However, the average data points from this experiment are plotted in figures presented in the next section. The tabulated results can be found in [11]. The most prominent issue with the 4MB file size was the errors that occurred for Case 2(d) and Case 2(e). These errors disappeared when the file size was reduced to 3MB. The assumption is that these errors are caused by the large file size and unknown memory allocation issues.

C. Summary of Analysis

To obtain a clearer picture of how the performance metrics compared to one another, plots were generated to include all ten cases and all file sizes. Figs. 2-4 were generated from data tables containing the latency and throughput results for the five message sizes (0.5MB-4MB). From Fig. 2, it can be seen that Case 1(b) and Case 2(b) continually had the worst performance for packet latency for the plaintext and secure cases, respectively. Since these two cases are for HISTORY=KEEP_ALL, this suggests the packet latency is most sensitive to the HISTORY QoS setting.

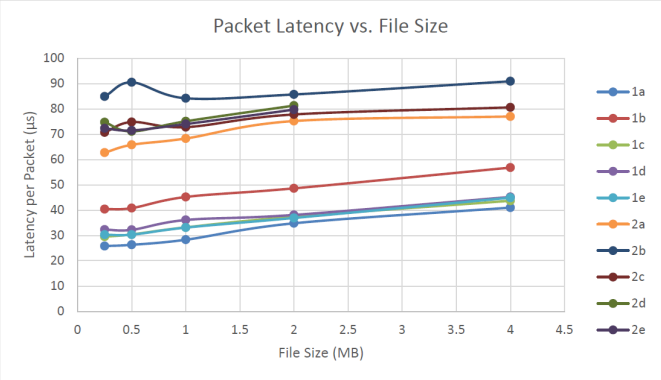

Fig. 2. Packet Latency vs. File Size plot for all cases

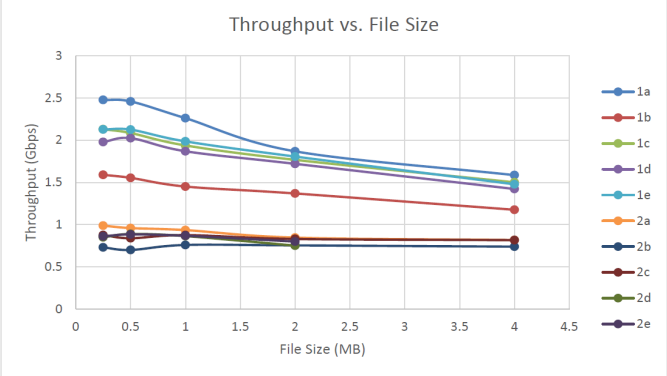

Fig. 4. Throughput vs. File Size for all cases

In Fig. 3, as message size increases so does the cost in terms of latency. As can be seen from Fig. 3, Cases 2(d) and 2(e) could not be evaluated beyond the 2MB file size (due to errors), but the trend shows poor performance. Lastly, throughput decreases as file size increases for all cases, as can be seen in Fig. 4.

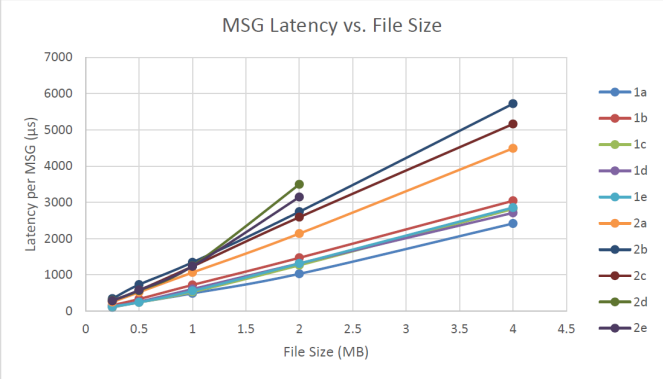

Fig. 3. MSG Latency vs. File Size plot for all cases

For all of the metrics illustrated in Figs. 2-4, security had a larger influence on performance than QoS configurations. Implementing security generates a typical increase in packet and message latency of greater than 140% compared to the baseline, while changes in the QoS profiles resulted in a increase typically less than 40%. Similarly, enabling security resulted in a minimum decrease in throughput of 48% across all message sizes, while QoS changes resulted in a maximum throughput decrease of 36% (often much lower). For system designers this suggests that where and how security is implemented in the system will have a large influence on performance, especially compared to the impact of QoS profiles.

V. Conclusion

This paper analyzed ROS 2 performance as it relates to latency, throughput, and overhead in a two node network. We have provided detailed insight on what tradeoffs are sacrificed as data is secured or different QoS profiles are used. Through our experimental setup, the capabilities and performance of ROS 2 were demonstrated through 50 different scenarios. The versatility of how the DDS security plugins can be applied is promising for use in actual warfare environments. In the future, ROS 2 is expected to include many more capabilities than it currently has. We plan to continue to examine ROS 2 network performance for UxS use cases as a function of network scale in lossy networks operating over wireless links.

Acknowledgements

This work was funded and sponsored by the Office of Naval Research via the Consortium for Robotics and Unmanned Systems Education and Research (CRUSER) at NPS.

References

[1] Office of the Secretary of Defense, "Unmanned systems integrated roadmap fy 2017-2042," 2017, https://www.defensedaily.com/wp-content/uploads/post_attachment/206477.pdf.

[2] Defense Science Board, "Task force report: The role of autonomy in DOD systems," 2012, https://fas.org/irp/agency/dod/dsb/autonomy.pdf.

[3] J. Kim, J.M. Smeraka, C. Cheung, S. Nepal, and M. Grobler, "Security and performance considerations in ROS 2: A balancing act," in *Computing Research Repository*, 2018.

[4] EPromisa, "FastRTPS documentation," https://readthedocs.org/projects/eprosima-fast-rtps/downloads/.

[5] EPromisa Middleware Experts, "ePromisa fast RTPS performance," https://www.eProsima.com/index.php/resources-all/performance/40-eprosimafast- rtps-performance.

[6] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in *Proc. of ACM International Conference on Embedded Software*, 2016, pp. 1–10.

[7] 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*, vol. 15, no. 3, pp. 1–15, 2018.

[8] G. Pardo and R. White, "Leveraging DDS security in ROS2," Presented at ROSCon 2018, 2018, https://roscon.ros.org/2018 /presentations/ROSCon2018_DDS_Security_in_ROS2.pdf.

[9] 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.

[10] Object Management Group, "The real-time publish subscriber protocol (RTPS) DDS interoperability wire protocol specification version 2.2," 2014, https://www.omg.org/spec/DDSI-RTPS/2.2.

[11] J. Fernandez, "Quality of service and cybersecurity communication protocols analysis for the robot operating system 2," M.S. thesis, Naval Postgraduate School, 2019.