



City Research Online

City St George's, University of London

Citation: Peeroo, K. (2026). Experimental and Model-Based Evaluation and Prediction of Data Distribution Service Performance. (Unpublished Doctoral thesis, City St George's, University of London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: <https://openaccess.city.ac.uk/id/eprint/36753/>

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to [City Research Online policy](#).

Experimental and Model-Based Evaluation and Prediction of Data Distribution Service Performance

Kaleem Peeroo

Doctor of Philosophy



School of Science and Technology
Department of Computer Science

January 2026

Contents

Contents	iii
Acknowledgements	ix
Abstract	xi
List of Figures	xiii
List of Tables	xxi
List of Acronyms	xxix
1 Introduction	1
1.1 Data Distribution Service	2
1.2 Research Objectives	6
1.3 Key Contributions	6
2 Literature Review	9
2.1 Introduction	9
2.2 Research Method	10
2.2.1 Research Question	11
2.2.2 Scope Identification	12
2.2.3 Search Strategy	12
2.2.4 Selection Criteria	13
2.2.5 Data Extraction	13
2.2.6 Data Summary	16
2.2.7 Critical Analysis and Theme Identification	17
2.3 Results	17
2.3.1 Overview of Selected Studies	18

2.3.2	Evaluation of Multiple Data Distribution Service Implementations	18
2.3.3	Evaluation of a Single Data Distribution Service Implementation	23
2.3.4	Evaluation of Unnamed Data Distribution Service Implementations . . .	28
2.3.5	Evaluation of Data Distribution Service Implementations in a Virtualised Environment	30
2.4	Discussion	30
2.4.1	General Discussion	30
2.5	Conclusion	32
3	Experimentation	35
3.1	Introduction	35
3.1.1	Contributions	36
3.2	Testbeds	36
3.2.1	Justification for Choice of Testbeds	36
3.2.2	Virtualised Testbed	37
3.2.3	Physical Testbeds: Raspberry Pi Clusters	37
3.3	Tools	39
3.3.1	RTI Perfctest	39
3.3.2	AutoPerf	41
3.3.3	AutoPerf Monitor	46
3.3.4	Performance Analysis Tool	47
3.4	Experimental Campaigns	49
3.4.1	Introduction	49
3.4.2	Justification for Choice of Experimental Parameters	49
3.4.3	Virtualised Environment	51
3.4.4	3 Raspberry Pi Cluster	54
3.4.5	5 Raspberry Pi Cluster	58
3.5	Data Processing	62
3.5.1	Raw Files to Datasets	62
3.5.2	Filtering Out Experiments	64
3.5.3	Transient and Outlier Analysis	65
3.6	Datasets	68
3.6.1	3 Raspberry Pi Cluster	68
3.6.2	5 Raspberry Pi Cluster	72

4	Experimental Performance Evaluation of Data Distribution Service	77
4.1	Introduction	77
4.1.1	Contributions	78
4.1.2	Structure of the Chapter	78
4.2	Virtualised Environments	78
4.2.1	Base Case	80
4.2.2	Multicast Exploration	81
4.2.3	Distributed Denial of Service	85
4.2.4	Malicious Subscriber Attack	89
4.2.5	Confirmation of Variation	91
4.2.6	Single versus Multiple Machine	92
4.3	Analysis on the Impact of Multicast	93
4.3.1	Introduction	93
4.3.2	Initial Analysis	95
4.3.3	Analysis on the 3 Raspberry Pi Multicast Dataset	100
4.3.4	Analysis on the 5 Raspberry Pi Multicast Dataset	110
4.3.5	Analysis on the 5 Raspberry Pi Quality of Service Dataset	115
4.3.6	Comparative Analysis of Latency Across Testbeds	123
4.3.7	Comparative Analysis of Throughput Across Testbeds	127
4.3.8	Quantile-Based Analysis	131
4.4	Analysis on the Impact of Data Length	152
4.4.1	Analysis Methodology	152
4.4.2	Latency	153
4.4.3	Throughput	168
4.4.4	Summary	180
4.5	Analysis on the Impact of Noise in the Network	181
4.5.1	3 Raspberry Pi Analysis	181
4.5.2	5 Raspberry Pi Analysis	182
4.5.3	Summary	183
4.6	Analysis of Faulty Throughput Measurements	184
4.6.1	Analysis	184
4.7	Limitations of the Experimental Setup	185
4.8	Conclusion	186

5	Model-Based Performance Prediction of Data Distribution Service	187
5.1	Introduction	187
5.1.1	Contributions	189
5.1.2	Background	190
5.2	Related Work	193
5.2.1	Our Contributions in Context	194
5.3	Datasets	195
5.3.1	Bandwidth Caps	195
5.4	Method	196
5.4.1	Dataset Processing	197
5.4.2	Bandwidth Cap Level Comparison	198
5.4.3	Dataset Allocation Strategy	198
5.4.4	Model Training	205
5.4.5	Summary	209
5.5	Hypotheses	209
5.5.1	Feasibility of Machine Learning for Data Distribution Service Performance Prediction	210
5.5.2	Interpolation Versus Extrapolation	210
5.5.3	Network Noise	211
5.5.4	Linear Versus Log Space	211
5.5.5	Single Versus Combined Output	211
5.6	Results	212
5.6.1	Initial Results	212
5.6.2	Interpolation Versus Extrapolation	221
5.6.3	Network Noise	224
5.6.4	Linear Versus Log Space	229
5.6.5	Single Versus Combined Output	241
5.7	Discussion	249
5.7.1	Initial discussion	249
5.7.2	Interpolation Versus Extrapolation	250
5.7.3	Network Noise	251
5.7.4	Linear Versus Log Space	252
5.7.5	Single Versus Combined Output	253

6 Conclusion and Future Work	257
6.1 Conclusion	257
6.2 Future Work	260
A Appendix A: Algorithmic Procedures	263
B Appendix B: Experimental Results	265
B.1 Participant Allocation Analysis on Multicast for 5Pi QoS Dataset	267
B.1.1 n Publishers and n Subscribers for all Participant Allocations	267
B.2 Data Length Quantile Analysis	282
B.2.1 Latency	282
B.2.2 Throughput	286
Bibliography	293

Acknowledgements

I want to express my most sincere gratitude and recognition to Vladimir Stankovic, Peter Popov, and Tillman Weyde for their valuable support throughout this work.

Abstract

The Data Distribution Service (DDS) specification is used in many real-time, mission-critical, and distributed systems e.g. Air Traffic Control (ATC), autonomous vehicle fleets, and medical devices. Setting up these systems and configuring the components and how they communicate is not easy because of the vast number of options that can be explored. While performance is a non-functional requirement for such systems, it plays a crucial role in the selection of configuration values. System designers want to select the configuration which produces the most ideal performance (lowest latency and highest throughput) throughout the run-time of the system.

To do this, one must run performance experiments for varied configurations to capture the behaviour of the system under different scenarios to find the most ideal configuration. This task is laborious and requires immense time, computational power, and effort.

We contributed to this problem with a two-pronged approach. First, we executed over 6,000 performance experiments on 3 different testbeds (both virtualised and physical), to evaluate the impact of various configuration parameters, such as Multicast (MC) and data length, on the performance of DDS in terms of latency and throughput, considered as the two most important performance metrics. Second, we use collected data to be the first to train various Machine Learning (ML) models (including Linear Regression (LR), Random Forest (RF), Extreme Gradient Boosting (XGB), and Feed-Forward Neural Network (FFNN)) and assess the effectiveness of ML for predicting the performance of DDS in terms of latency and throughput, as characterised by their distributions.

Our results uncover counter-intuitive findings related to MC, insights into how larger data lengths outperform smaller data lengths in terms of latency, specifically in worst-case scenarios, suitable demonstration of the applicability of ML with accuracies up to 0.99 for interpolation and several cases with accuracies above 0.9 for extrapolation, counter-intuitive findings showing that introducing noise via a bandwidth throttle improves the predictive ability of the ML models, and more.

List of Figures

1.1	Basic DDS Architecture with Publisher, Subscriber, DataWriter, DataReader, and Topic	3
1.2	Simplified DDS Architecture Stack	3
3.1	Physical 5-Node RPi Cluster Testbed	39
3.2	Latency and Throughput Test Comparison in RTI's Perftest.	41
3.3	Modular class architecture of the AutoPerf framework.	44
3.4	The AutoPerf experimental lifecycle and data flow.	45
3.5	Example of Transient Analysis with Binned Averages for 25 Random Experiments	67
3.6	Pairplot of Configuration Parameters' values for 3Pi ML Without a Bandwidth Cap split between "Val" (used for validating interpolation and shown in green) versus "Out" (used for validating extrapolation and shown in red).	70
3.7	Pairplot of Configuration Parameters' values for 3Pi ML with Low Bandwidth Cap (6 Mbps) split between "Val" (used for validating interpolation and is shown in green) versus "Out" (used for validating extrapolation and is shown in red).	71
3.8	Pairplot of Configuration Parameters for 3Pi ML with High Bandwidth Cap (8 Mbps) split between "Val" (used for validating interpolation and is shown in green) versus "Out" (used for validating extrapolation and is shown in red).	72
3.9	Pairplot of Configuration Parameters for 5Pi ML Without a Bandwidth Cap split between "Val" (used for validating interpolation and is shown in green) versus "Out" (used for validating extrapolation and is shown in red).	74
3.10	Pairplot of Configuration Parameters for 5Pi ML with Low Bandwidth Cap (24 Mbps) split between "Val" (used for validating interpolation and is shown in green) versus "Out" (used for validating extrapolation and is shown in red).	75
3.11	Pairplot of Configuration Parameters for 5Pi ML with High Bandwidth Cap (71 Mbps) split between "Val" (used for validating interpolation and is shown in green) versus "Out" (used for validating extrapolation and is shown in red).	76

4.1	Latency measurements for the base case campaign, comparing UC and MC communication over time across 3 runs.	80
4.2	eCDF of throughput for the base case campaign, comparing UC and MC communication across all runs across each individual subscriber.	81
4.3	Box plots comparing latency for UC in green and MC in red communication across different configurations in the MC exploration campaign.	82
4.4	eCDFs (left column) and histograms (right column) of latency for UC (green) and MC (red) communication across 3, 10, 25, and 75 publishers/subscribers in the MC exploration campaign. 95% confidence intervals included.	83
4.5	Box plots comparing throughput for UC in green and MC in red communication across different configurations in the MC exploration campaign.	84
4.6	eCDFs (left column) and histograms (right column) of throughput for UC (green) and MC (red) communication across 3, 10, 25, and 75 publishers/subscribers in the MC exploration campaign. 95% confidence intervals included.	85
4.7	Average latency for the DDoS campaign with 25 publishers and 25 subscribers, comparing UC and MC communication across different data lengths.	87
4.8	eCDFs with 95% confidence intervals (left column) and histograms (right column) of latency (ms) for normal (green) and malicious (red) cases with 10, 25, 50, and 75 publishers.	89
4.9	eCDF and PDF of throughput for malicious subscribers across different subscriber counts both individually and combined.	90
4.10	eCDFs with 95% confidence intervals for UC (green) and MC (red) latency (ms) of three 15-minute runs (left) versus three 24-hours runs (right).	91
4.11	Comparing variation confirmation eCDF against base case for both latency and throughput. Top: latency. Bottom: throughput. Left: base case. Right: variation confirmation.	92
4.12	Latency over time for single (top) versus multiple machines (bottom) over 3 runs combined.	93
4.13	Latency eCDFs for varying participant numbers.	96
4.14	Throughput eCDFs for varying participant numbers.	98
4.15	eCDFs of MC (red) and UC (green) latency for 1, 5, 10, 15, and 25 publishers and subscribers for the 3Pi MC dataset. The x-axis is in log scale.	102
4.16	eCDFs of MC (red) and UC (green) throughput for 1, 5, 10, 15, and 25 publishers and subscribers for the 3Pi MC dataset.	103

4.17	eCDFs of MC (red) and UC (green) latency for 1 publisher and 1, 5, 10, 15, and 25 subscribers for the 3Pi MC dataset. The x-axis is in log scale.	105
4.18	eCDFs of MC (red) and UC (green) throughput for 1 publisher and 1, 5, 10, 15, and 25 subscribers for the 3Pi MC dataset.	106
4.19	eCDFs of MC (red) and UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers for the 3Pi MC dataset. The x-axis is in log scale.	107
4.20	eCDFs of MC (red) and UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers for the 3Pi MC dataset.	108
4.21	eCDFs of MC (red) and UC (green) latency for 1, 5, 10, and 15 publishers and subscribers for the 5Pi MC dataset. The x-axis is in log scale.	111
4.22	eCDFs of MC (red) and UC (green) throughput for 1, 5, 10, and 15 publishers and subscribers for the 5Pi MC dataset.	112
4.23	eCDFs of MC (red) and UC (green) latency for 1 publisher and 1, 5, 10, 15, 20, and 25 subscribers for the 5Pi MC dataset. The x-axis is in log scale.	113
4.24	eCDFs of MC (red) and UC (green) throughput for 1 publisher and 1, 5, 10, 15, 20, and 25 subscribers for the 5Pi MC dataset.	114
4.25	eCDFs of MC (red) and UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 20 publishers for the 5Pi MC dataset. The x-axis is in log scale.	116
4.26	eCDFs of MC (red) and UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 20 publishers for the 5Pi MC dataset.	117
4.27	eCDFs of MC (red) and UC (green) latency for 1, 5, and 10 publishers and subscribers for the 5Pi QoS dataset. The x-axis is in log scale.	119
4.28	eCDFs of MC (red) and UC (green) throughput for 1, 5, and 10 publishers and subscribers for the 5Pi QoS dataset.	120
4.29	eCDFs of MC (red) and UC (green) latency for 1 publisher and 1, 5, 10, and 25 subscribers for the 5Pi QoS dataset. The x-axis is in log scale.	121
4.30	eCDFs of MC (red) and UC (green) throughput for 1 publisher and 1, 5, 10, and 25 subscribers for the 5Pi QoS dataset.	122
4.31	eCDFs of MC (red) and UC (green) latency for 1 subscriber and 1, 5, 10, and 25 publishers for the 5Pi QoS dataset. The x-axis is in log scale.	124
4.32	eCDFs of MC (red) and UC (green) throughput for 1 subscriber and 1, 5, 10, and 25 publishers for the 5Pi QoS dataset.	125
4.33	Histogram of data distribution for the 0 to 5 percentile range.	133
4.34	Histogram of data distribution for the 5 to 95 percentile range.	133

4.35	Histogram of data distribution for the 95 to 100 percentile range.	135
4.36	Histogram of data distribution for the 0 to 100 percentile range.	135
4.37	Histogram of data distribution for the 0 to 5 percentile range in throughput analysis.	145
4.38	Histogram of data distribution for the 5 to 95 percentile range in throughput analysis.	145
4.39	Histogram of data distribution for the 95 to 100 percentile range in throughput analysis.	147
4.40	Histogram of data distribution for the 0 to 100 percentile range in throughput analysis.	147
4.41	Histogram showing distribution of best performing data lengths per configuration parameter between 0 and 5 percentiles for comparison of 100 B and 1 KB. . . .	155
4.42	Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 100 B and 1 KB. . . .	155
4.43	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B and 1 KB. . .	156
4.44	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B and 512 B. . .	159
4.45	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B, 512 B, and 1 KB.	162
4.46	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles.	165
4.47	Histogram showing distribution of best performing data lengths per configuration parameter between 0 and 5 percentiles for comparison of 100 B and 1000 B. . .	171
4.48	Histogram showing distribution of best performing data lengths per configuration parameter between 0 and 100 percentiles for comparison of 100 B and 1000 B. . .	171
4.49	Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 100 B and 1000 B. . .	172
4.50	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B and 1000 B. . .	172
4.51	Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 100 B, 512 B, and 1 KB.	176

4.52	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B, 512 B, and 1 KB.	176
4.53	eCDFs with 95% confidence intervals of latency (ms) for 6 Mbps (green) versus 8 Mbps (red) bandwidth throttle on 3Pi testbed. x-axis is log-scaled.	181
4.54	eCDFs with 95% confidence intervals of throughput (Mbps) for 6 Mbps (green) versus 8 Mbps (red) bandwidth throttle on 3Pi testbed.	182
4.55	eCDFs with 95% confidence intervals of latency (ms) for 71 Mbps (red) versus 24 Mbps (green) bandwidth throttle on 5Pi testbed.	182
4.56	eCDFs with 95% confidence intervals of throughput (Mbps) for 71 Mbps (red) versus 24 Mbps (red) bandwidth throttle on 5Pi testbed.	183
5.1	3D scatter plot of the values for publisher/subscriber count and data length (B).	199
5.2	All variations of the 3D scatter plot as seen in 2D.	200
5.3	All variations of the 3D scatter plot as seen in 2D (green area indicates values used for interpolation and blue area indicates values used for extrapolation).	202
5.4	Scatter plots of predicted values on y-axis and actual values on x-axis for 9 randomly picked tests using the RFs model to extrapolate for latency.	219
5.5	Input Distributions for Validation and Out-of-Sample Datasets for 5Pi 24 Mbps Dataset	223
A.1	Sketch of the algorithmic procedure for eCDF stochastic ordering. The left most and right most values are calculated per pairing. The value that appear left-/right-most majority of the time without intersecting with others is classified as the most performant.	263
B.1	Latency comparison between the 15-minute and 6-hour configurations. Top row: latency PDF. Bottom row: latency eCDF. Left column: 6 hour run. Right column: Latencies averages from 3 15-minute runs.	266
B.2	Box plots comparing MC (red) to UC (green) latency for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.	267
B.3	Box plots comparing MC (red) to UC (green) throughput for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.	267

B.4	Box plots comparing MC (red) to UC (green) latency for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.	268
B.5	Box plots comparing MC (red) to UC (green) throughput for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.	268
B.6	Box plots comparing MC (red) to UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 3Pi Multicast Exploration.	269
B.7	Box plots comparing MC (red) to UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 3Pi Multicast Exploration.	269
B.8	Box plots comparing MC (red) to UC (green) latency for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.	270
B.9	Box plots comparing MC (red) to UC (green) throughput for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.	270
B.10	Box plots comparing MC (red) to UC (green) latency for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.	271
B.11	Box plots comparing MC (red) to UC (green) throughput for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.	271
B.12	Box plots comparing MC (red) to UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi Multicast Exploration.	272
B.13	Box plots comparing MC (red) to UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi Multicast Exploration.	272
B.14	Box plots comparing MC (red) to UC (green) latency for 1, 5, 10, and 15 publishers and subscribers. The y-axis is in log scale. From 5Pi QoS.	273
B.15	Box plots comparing MC (red) to UC (green) throughput for 1, 5, 10, and 15 publishers and subscribers. The y-axis is in log scale. From 5Pi QoS.	273
B.16	Box plots comparing MC (red) to UC (green) latency for 1 publisher and 1, 5, 10, and 25 subscribers. The y-axis is in log scale. From 5Pi QoS.	274

B.17	Box plots comparing MC (red) to UC (green) throughput for 1 publisher and 1, 5, 10, and 25 subscribers. The y-axis is in log scale. From 5Pi QoS.	274
B.18	Box plots comparing MC (red) to UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi QoS.	275
B.19	Box plots comparing MC (red) to UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi QoS.	275
B.20	eCDFs comparing UC (green) to MC (red) latency for the “1P3S” allocation. From 5Pi QoS.	276
B.21	eCDFs comparing UC (green) to MC (red) throughput for the “1P3S” allocation. From 5Pi QoS.	276
B.22	eCDFs comparing UC (green) to MC (red) latency for the “2P2S” allocation. From 5Pi QoS.	277
B.23	eCDFs comparing UC (green) to MC (red) throughput for the “2P2S” allocation. From 5Pi QoS.	277
B.24	eCDFs comparing UC (green) to MC (red) latency for the “3P1S” allocation. From 5Pi QoS.	277
B.25	eCDFs comparing UC (green) to MC (red) throughput for the “3P1S” allocation. From 5Pi QoS.	278
B.26	Multicast analysis using the 5Pi QoS Capture Dataset looking at different participant allocations.	279
B.27	Multicast analysis using the 5Pi QoS Capture Dataset looking at all combinations of publishers and subscribers	281
B.28	Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 512 B and 1 KB. . . .	289
B.29	Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 512 B and 1 KB. . .	289

List of Tables

2.1	Parameters and their descriptions gathered for all papers in the literature review on the experimental performance evaluation of DDS.	15
2.2	Default QoS parameter values.	24
3.1	Summary of Testbed Specifications	37
3.2	MC Exploration Campaign Configuration. Latency count is the number of messages to send before a latency measurement is taken. In this case latency measurements are taken every 1000 messages.	52
3.3	Malicious Subscriber Attack Campaign Configuration	52
3.4	3Pi QoS Capture Campaign Configuration	54
3.5	Participant Threshold Campaign Configuration	55
3.6	3Pi Multicast Exploration Campaign Configuration	56
3.7	Low BW Cap Campaign Configuration (6 Mbps)	57
3.8	High BW Cap Campaign Configuration (8 Mbps)	58
3.9	QoS Capture Configuration for 1P 3S, 2P 2S, and 3P 1S	58
3.10	Increasing Load Campaign Configuration for 1P1S, 1P2S, and 1P3S	59
3.11	5Pi Participant Threshold Campaign Configuration	60
3.12	5Pi Multicast Exploration Campaign Configuration	60
3.13	5Pi Noise Generation Campaign Configuration	61
3.14	ML Data Collection Campaign Configuration	61
3.15	Experiment Counts for 3Pi Datasets.	68
3.16	Transient Percentages for 3Pi Datasets.	69
3.17	Summary Statistics for the Configuration Parameter Values for 3Pi ML Without a Bandwidth Cap.	69
3.18	Configuration Parameters for 3Pi ML with Low Bandwidth Cap (6 Mbps). . . .	70
3.19	Configuration Parameters for 3Pi ML with High Bandwidth Cap (8 Mbps). . . .	71

3.20	Summary of experiment counts and truncation percentages for latency and throughput in the 5Pi Quality of Service (QoS) Capture and Multicast Exploration campaigns.	73
3.21	Configuration Parameters for 5Pi ML Without a Bandwidth Cap.	73
3.22	Configuration Parameters for 5Pi ML with Low Bandwidth Cap (24 Mbps). . .	73
3.23	Configuration Parameters for 5Pi ML with High Bandwidth Cap (71 Mbps). . .	74
4.1	Mean differences in latency for UC and MC communication during the DDoS campaign across different data lengths.	85
4.2	Mean differences in latency for UC and MC communication during the DDoS campaign across different data lengths, expressed as percentages.	88
4.3	Most performant communication type per participant configuration for both latency and throughput on the VM testbed.	99
4.4	Most performant type of communication (UC or MC) in terms of latency and throughput for each combination of publisher and subscriber count according to visual stochastic ordering seen when comparing eCDFs. Mostly UC/MC indicates that there was some intersection seen in the eCDF. None indicates that there was no clear difference between the 2 communication types. Values in bold indicate cases with the same outcomes as seen in VMs.	109
4.5	Comparison of Latency and Throughput for Different Publisher and Subscriber Counts. Most performant communication type (UC or MC) per participant count determined by stochastic ordering seen in eCDF. “Mostly MC” indicates that the majority of the eCDF showed stochastic ordering in favour of MC with some intersection. Results also seen from the 3Pi analysis as well as here are highlighted in bold. “none” indicates that there was no clear difference between the 2 communication types.	118
4.6	Comparison of Latency and Throughput for Different Publisher and Subscriber Counts for the 5Pi QoS dataset. Most performant communication type (UC or MC) per participant count determined by stochastic ordering seen in eCDF. “Mostly MC” indicates that the majority of the eCDF showed stochastic ordering in favour of MC with some intersection.	126

4.7	Most performant communication type (UC or MC) according to latency for each variation of publisher and subscriber count summarised across all datasets on physical testbeds. Values in bold represent results consistent across all testbeds. “-” indicates that the configuration was not tested in that dataset.	127
4.8	Most performant communication type (UC or MC) according to throughput for each variation of publisher and subscriber count summarised across all datasets on physical testbeds. Values in bold represent results consistent across all testbeds. “-” indicates that the configuration was not tested in that dataset.	130
4.9	Distribution of cases across percentile ranges for UC and MC communication (“Uncategorised” indicates no stochastic ordering).	132
4.10	Summary of case classifications across percentile ranges for latency analysis. . .	138
4.11	Statistical analysis for the 0 to 5 percentile range. Bold values indicate statistical significance (p-value < 0.05).	140
4.12	Statistical analysis for the 5 to 95 percentile range. Bold values indicate statistical significance (p-value < 0.05).	140
4.13	Statistical analysis for the 95 to 100 percentile range. Bold values indicate statistical significance (p-value < 0.05).	141
4.14	Statistical analysis for the 0 to 100 percentile range.	142
4.15	Summary of V values for statistically significant parameters across percentile ranges in latency analysis.	143
4.16	Distribution of cases across percentile ranges for MC communication in throughput analysis.	143
4.17	Summary of case classifications across percentile ranges for throughput analysis.	149
4.18	V statistics for throughput analysis across percentile ranges. Highlighted values correspond to statistically significant parameters (p-value < 0.05).	150
4.19	Chi-square and p-value statistics for throughput analysis across percentile ranges.	151
4.20	Counts of best performing data lengths for latency between 100 B and 1 KB. . .	153
4.21	Counts of best performing data lengths for latency between 100 B and 512 B. .	157
4.22	Counts of best performing data lengths for latency between 100 B, 512 B, and 1 KB.	160
4.23	Counts of best performing data lengths for latency between 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B. Only data lengths with a least 1 case of best performance are shown.	163

4.24	Summary of all statistically significant relationships between parameter and data lengths for latency performance.	166
4.25	Counts of best performing data lengths for throughput between 100 B and 512 B. 100 B did not outperform 512 B in any case.	168
4.26	Counts of best performing data lengths for throughput between 100 B and 1 KB.	169
4.27	V correlation analysis for throughput between 100 B and 1 KB. Highlighted values indicate statistical significance (p -value < 0.05).	173
4.28	χ^2 correlation analysis (0–95 percentiles). Highlighted values indicate statistical significance ($p < 0.05$).	173
4.29	χ^2 correlation analysis (95–100 and overall). Highlighted values indicate statistical significance ($p < 0.05$).	173
4.30	Counts of best performing data lengths for throughput between 100 B, 512 B, and 1 KB. 100 B did not outperform the other data lengths in any case.	174
4.31	V correlation analysis for throughput between 100 B, 512 B, and 1 KB. Highlighted values indicate statistical significance (p -value < 0.05).	177
4.32	χ^2 correlation analysis (0–95 percentiles) for throughput between 100 B, 512 B, and 1 KB.	177
4.33	χ^2 correlation analysis (95–100 and overall) for throughput between 100 B, 512 B, and 1 KB.	177
4.34	Counts of best performing data lengths for throughput between 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B. Only data lengths that outperformed all others are shown.	177
4.35	Summary of statistically significant relationships between parameters and data lengths for throughput performance.	178
5.1	Summary of the number of samples per testbed and bandwidth cap.	196
5.2	Summary of the configuration parameters values used to split the dataset into inside (used for interpolation) and outside (used for extrapolation) samples for 3Pi.	204
5.3	Summary of the configuration parameters values used to split the dataset into inside (used for interpolation) and outside (used for extrapolation) samples for 5Pi.	205
5.4	Optuna hyperparameter search space for RF, XGB, and FFNN models.	207

5.5	R ² values during training and testing for LR and RF for interpolation (int) shown in green and extrapolation (ext) shown in blue.	213
5.6	RMSE values during training and testing for LR and RF for interpolation (int) as shown in green and extrapolation (ext) as shown in blue.	214
5.7	Interpolation (int) versus extrapolation (ext) for LR focusing on the R^2 metric. Rows with positive percentage differences are highlighted in bold.	215
5.8	Interpolation (int) versus extrapolation (ext) for LR focusing on the RMSE metric. Negative values are highlighted in bold.	216
5.9	Interpolation (int) versus extrapolation (ext) for RFs focusing on the R^2 metric.	217
5.10	Interpolation (int) versus extrapolation (ext) for RFs focusing on the RMSE metric. Negative values are highlighted in bold indicating improvement with extrapolation.	220
5.11	Specific Model Configurations Where Extrapolation R^2 is Greater Than or Equal to 0.8	222
5.12	Cases where Extrapolation Outperforms Interpolation	223
5.13	Descriptive Statistics for Mean R^2 Values Across BW Cap Levels	225
5.14	Counts of Lowest and Highest Mean/Median R^2 by BW Cap Level	226
5.15	Lowest and Highest Mean R^2 Counts by Testbed and BW Cap Level	226
5.16	Lowest and Highest Mean R^2 Counts by Metric and BW Cap Level	227
5.17	Lowest and Highest Mean R^2 Counts by Model and BW Cap Level	228
5.18	Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for LR Models. Bigger value indicates the transformation that yields a higher R^2 score.	231
5.19	Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for FFNN Models. Bigger value indicates the transformation that yields a higher R^2 score.	231
5.20	Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for RF Models. Bigger value indicates the transformation that yields a higher R^2 score.	232
5.21	Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for XGB Models. Bigger value indicates the transformation that yields a higher R^2 score.	232
5.22	Cases where Logarithmic Transformation Outperforms Linear Transformation for Worst-Case Percentiles for Latency (all cases were 3Pi No Cap)	233

5.23 Latency R^2 Comparison: Linear vs. Logarithmic Transformations for All Percentiles Sorted by Dataset	234
5.24 Cases where Linear Transformation Outperforms Logarithmic Transformation for All Percentiles for Latency	235
5.25 Cases where Logarithmic Transformation Outperforms Linear Transformation for All Percentiles for Latency	235
5.26 Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for FFNN Models	236
5.27 Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for LR Models	236
5.28 Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for RF Models	237
5.29 Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for XGB Models	237
5.30 Throughput R^2 Comparison: Logarithmic vs. Linear Transformations for Normal vs. Worst Case Percentiles	238
5.31 Number of cases where Linear vs. Logarithmic Transformations are Most Performant for Throughput R^2 by Model	238
5.32 Number of cases where Linear vs. Logarithmic Transformations are Most Performant for Throughput R^2 by Dataset	239
5.33 Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for All Percentiles	239
5.34 Cases where Linear Transformation Outperforms Logarithmic Transformation for All Percentiles (Throughput)	240
5.35 Cases where combined output outperforms single output for latency.	242
5.36 Cases where Single Output Outperforms Combined Output for Latency	244
5.37 Cases where single output outperforms combined output for throughput.	245
B.1 Summary of 100 B performance against other data lengths.	283
B.2 Summary of 512 B performance against other data lengths.	285
B.3 χ^2 correlation analysis for throughput between 100 B and 512 B.	287
B.4 V correlation analysis for throughput between 100 B and 512 B.	288
B.5 Counts of best performing data lengths for throughput between 512 B and 1 KB.	288
B.6 χ^2 correlation analysis for throughput between 512 B and 1 KB.	291

B.7	V correlation analysis for throughput between 512 B and 1 KB.	292
-----	---	-----

List of Acronyms

R^2 Coefficient of Determination.

χ^2 Chi-squared.

μs microseconds.

ACK Acknowledgement.

AI Artificial Intelligence.

AMQP Advanced Message Queuing Protocol.

ANOVA Analysis of Variance.

ATC Air Traffic Control.

B byte.

BW Bandwidth.

CLI Command Line Interface.

CPU Central Processing Unit.

DCPS Data-Centric Publish-Subscribe.

DDoS Distributed Denial of Service.

DDS Data Distribution Service.

DDSI DDS Interoperability wire protocol.

DF DataFrame.

DLRL Data Local Reconstruction Layer.

DoF Degrees of Freedom.

eCDF Empirical Cumulative Distribution Function.

ESS Experiment Status Spreadsheet.

FFNN Feed-Forward Neural Network.

GAN Generative Adversarial Network.

GB gigabyte.

H Kruskal-Wallis.

HLA High-Level Architecture.

HSD Tukey's Honest Significant Difference.

IGMP Internet Group Management Protocol.

IP Internet Protocol.

KB kilobyte.

KS Kolmogorov-Smirnov.

LAN Local Area Network.

LR Linear Regression.

MAE Mean Absolute Error.

MaxAE Maximum Absolute Error.

MB megabyte.

MBps megabytes per second.

Mbps megabits per second.

MC Multicast.

MEP Multicast Exhaustion Point.

ML Machine Learning.

MQTT Message Queuing Telemetry Transport.

MRP Multicast Routing Protocol.

ms milliseconds.

NACK Negative Acknowledgement.

OMG Object Management Group.

PAT Performance Analysis Tool.

PCG Preset Combination Generation.

PDF Probability Density Function.

QoS Quality of Service.

RAM Random Access Memory.

RCG Random Combination Generation.

RF Random Forest.

RMSE Root Mean Square Error.

RPi Raspberry Pi.

RRMSE Relative Root Mean Square Error.

RTI Real-Time Innovations.

RTT Round Trip Time.

SPN Stochastic Petri Net.

SPT Shortest Path Tree.

SSH Secure Shell.

TCP Transmission Control Protocol.

TOML Tom’s Obvious, Minimal Language.

U Mann-Whitney U.

UC Unicast.

UDP User Datagram Protocol.

V Cramér’s V.

vLAN Virtual Local Area Network.

VM Virtual Machine.

W Shapiro-Wilk.

WAN Wide Area Network.

XGB Extreme Gradient Boosting.

ZMQ ZeroMQ.

Chapter 1

Introduction

Distributed mission-critical systems use implementations of the DDS specification as defined by Object Management Group (OMG) (*Object Management Group* n.d.) to communicate data amongst internal system entities in real-time. These use cases are very different. Some systems require an isolated Local Area Network (LAN) e.g. robots while others require a virtualised Wide Area Network (WAN) setup e.g. distributed simulations. Different systems have different performance requirements and are deployed on different physical infrastructures. These result in different configurations of DDS being optimal for different systems. Identifying a good-enough configuration is non-trivial because of the absence of the configuration's performance, which requires performance testing, and will unlikely identify the most performant configurations, as such a discovery would require exhaustive performance testing of numerous configurations.

Testing out multiple configurations has a substantial temporal, energy, and financial cost. This cost is multiplied when one requires thorough investigation since one would ideally run the performance tests for as long as possible to accurately capture the behaviour during run-time. Deploying this on the desired physical infrastructure which would be powered during the performance test, costs energy, and ensuring that the system is started from a clean slate after each performance test costs manual effort.

Our initial contribution to this problem involves the collection of performance data over a wide variety of distinct configurations achieved by first deploying three testbeds (one virtualised and two physical) that mimic real-life use cases. The benchmarking tool provided by Real-Time Innovations (RTI) (*RTI* n.d.) (the largest DDS vendor at the time) was extended to facilitate this. Relationships between the configuration parameters and their impact on DDS performance were identified to help system designers in their decision making process when deploying DDS systems and is especially useful when the performance of a configuration

parameter value steers from expectations as seen in our results.

We contribute further to this problem by introducing a novel data-driven method for predicting the performance of unseen configurations lowering temporal, energy, and financial costs. The ideal scenario involves the system designer running performance tests on a smaller subset of the configuration space and using this data to calibrate the pre-trained (trained by our extensive performance data) ML models to predict the performance of untested configurations tailored to the system being designed. Since this has not been previously accomplished, we start by using LR as a baseline to be compared with RF, XGB, and FFNN models. We then extend the work to explore different modelling techniques specific to this problem to provide insights into how to better model the performance of DDS.

To this end, this research is guided by the following questions:

1. What are the quantitative impacts of key QoS parameters, multicasting, data lengths, and network noise on DDS latency and throughput?
2. To what extent can ML models accurately predict DDS performance for untested configurations, and which models are most effective?
3. How do network topology and noise affect DDS performance, and can these effects be accurately modelled and predicted?

This introductory chapter aims to provide a concise overview of the thesis structure. It also furnishes the requisite foundational knowledge concerning DDS for a comprehensive understanding of the subsequent chapters.

This thesis is organised into six chapters. Chapter 1 provides an introduction to the research and outlines the thesis structure. Chapter 2 presents the background and a comprehensive literature review pertinent to DDS performance evaluation and all literature related to performance prediction is found in Chapter 5. Chapter 3 details the experimental methodology, testbed configurations, and data collection processes. Chapter 4 presents the experimental performance evaluation of DDS, analysing the impact of various parameters. Chapter 5 focuses on the development and evaluation of model-based performance prediction techniques. Finally, Chapter 6 concludes the thesis and outlines avenues for future research.

1.1 Data Distribution Service

DDS is a widely adopted specification developed by the OMG. It is primarily employed in real-time and mission-critical distributed systems. Such systems include, but are not limited

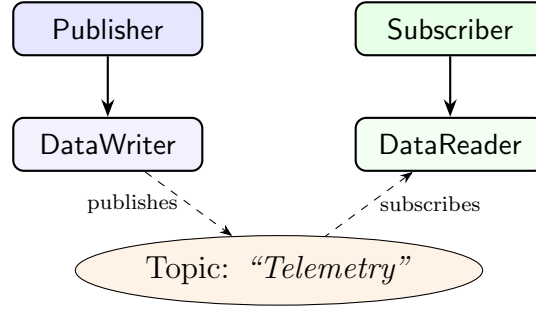


Figure 1.1: Basic DDS Architecture with Publisher, Subscriber, DataWriter, DataReader, and Topic

to, autonomous vehicles, energy management systems, and air traffic control. For the empirical investigations presented herein, the RTI Connexst DDS implementation was used. Figure 1.1 illustrates a simplified architecture of DDS. DDS facilitates seamless communication between multiple system components through the establishment of logical communication channels known as *topics*. Communication within DDS is mediated by two primary entity types: publishers and subscribers. Publishers use *DataWriters* to generate and transmit messages on a designated *topic*. Conversely, subscribers employ *DataReaders* to acquire and process messages from the specified *topics*. A single *publisher* may operate multiple *DataWriters*, and similarly, a *subscriber* may manage several *DataReaders*. The architecture of DDS inherently supports the dynamic addition or removal of *publishers* and *subscribers* in real-time following system deployment, thereby demonstrating its inherent scalability. The most common terms used throughout DDS are *samples*, *instances*, *topics*, and *groups*. A *topic* is a uniquely named data type definition, an *instance* refers to an instance of a topic, a *group* refers to all instances belonging to a DataWriter/DataReader, and a *sample* is an individual item of an instance of a topic which is used interchangeably with message.

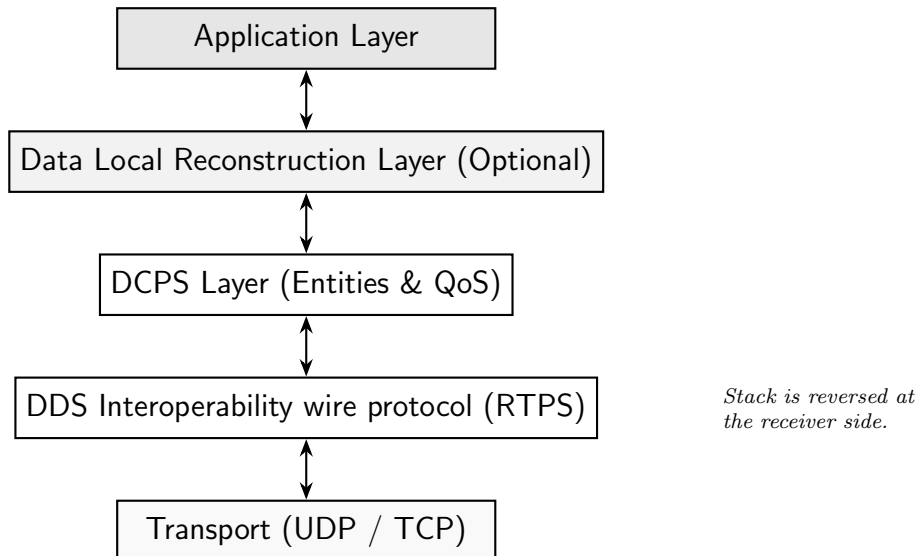


Figure 1.2: Simplified DDS Architecture Stack

As depicted in Figure 1.2, the DDS architecture consists of two primary layers, with an optional third layer. The first layer, called the Data Local Reconstruction Layer resides on the same level as the applications and is optional. This layer allows for the ease of integration of DDS with applications. The lower level is called Data-Centric Publish-Subscribe (DCPS) layer which is where all the communication happens. In this layer all the participants, both publishers and subscribers, reside and within them are their constituent DataWriters and DataReaders. The general process starts within the application where the data is constructed and sent to the DCPS layer where the publishers publish this data. This data is then transferred to the DDS Interoperability wire protocol where the data is then further transported using User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) connections where it is received by another machine and this “stack” is reversed to be used by subscriber(s).

The DDS specification has an extensive range of QoS parameters allowing for numerous configurations specifically customised for various scenarios. These QoS parameters range from data length of the DDS messages being sent throughout the communications, to the history parameter which decides where the participants store packets that have already been sent/received. The DDS specification states that there are 22 parameters of various data types leading to an abundant number of parameter value combinations - too many for one to try out every single combination. If we assume each of the 22 parameters could take a binary value, there is a total of 2^{22} combinations which is over four million. In fact, the actual number of possible combinations is likely much greater as many parameters assume more than two possible values. This is true despite some combinations of parameter values being incompatible, which will decrease the number of possible combinations, and perhaps drastically so. Moreover, there are non-QoS parameters with unlimited values such as data lengths of the messages sent between publishers and subscribers, number of publishers, or number of subscribers.

Given the vastness of this parameter space, the experimental and modelling work presented in this thesis focuses on a critical subset of these parameters, identified for their significant impact on performance and their configurability. These include UC/MC communication, Data Length, Number of Publishers, Number of Subscribers, Durability, and Reliability.

UC and MC are communication methods and are not considered part of the DDS specification. UC represents the communication between two endpoints where there is one sender and one receiver whilst multicast represents the communication from one endpoint to multiple endpoints (but not all endpoints - i.e. not broadcast). Theoretically, using the MC communication method should increase the throughput as an increased amount of data is being disseminated. By default, DDS does not use MC communication though this non-standard QoS parameter

can be easily configured.

Data length represents the size of the messages sent between publishers and subscribers and is measured in B. Increasing the data length should theoretically increase both the latency and throughput and various vendor implementations of DDS may use different default values for this parameter.

The number of publishers and the number of subscribers are self-explanatory. Increasing the number of publishers should theoretically increase the throughput depending on the message length and the size of the bandwidth since an increased number of messages are delivered to the subscribers. Similarly, increasing the number of subscribers should theoretically also result in a related increase.

Durability controls whether readers that join late can access the data that has previously been published. It can take one of four values including **volatile**, **transient_local**, **transient**, and **persistent**. The **volatile** value does not save or deliver formerly published samples. The **transient_local** value saves and delivers published samples if the DataWriter that published the samples still exists. The **transient** value saves and delivers published samples using a memory-based service whilst the **persistent** value saves and delivers old samples using a disk-based service. The impact on the performance of DDS can be affected by the durability since extra packets will either be propagated or not. By default, the Durability QoS parameter assumes the value of **volatile**.

The Reliability parameter decides whether missed samples are sent again or not. The parameter can take the value of **best effort** or **reliable** (the latter is set by default). **Best effort** does not resend any missed samples and the order of the samples in the history of the DataWriter may not match the history of received samples on the DataReader. On the other hand, if the value of the parameter is set to **reliable**, then any missed samples are resent. If the DataReader does not receive a sample, then the DataWriter blocks the samples ready to be sent to solve the issue. This means that the DataWriter will block all future samples while it attempts to resend the missed sample. The time period elapsed during this blocking can also be configured in order to determine how long the DataWriter can afford to block for. Latency should be affected depending on the value set (**best effort** or **reliable**).

Another feature of DDS is the use of domains. Domains allow participants to be grouped together and communicate with each other. Participants on the same domain can communicate with each other, but participants on different domains cannot. This is useful for isolating different applications or components within a system, ensuring that they do not interfere with each other.

1.2 Research Objectives

Building upon the identified challenges in DDS performance assessment, this thesis pursues several key research objectives. The primary aim is to conduct a comprehensive empirical investigation into the performance characteristics of DDS under a wide array of operational conditions. Specifically, this involves quantifying the impact of diverse QoS parameters, data lengths, and network environmental factors on critical performance metrics such as latency and throughput. A further objective is to ascertain the feasibility and efficacy of employing ML methodologies to accurately predict DDS performance for configurations not subjected to direct empirical testing. Ultimately, this research seeks to provide actionable insights and a robust predictive framework to facilitate the optimal design and deployment of DDS-based real-time distributed systems.

1.3 Key Contributions

This thesis makes several novel and significant contributions to the field of DDS performance evaluation and prediction.

- **Testbeds:** This research has developed three distinct testbeds (one virtualised, and two comprising of RPIs) used in the data collection stage to gather performance data of DDS.
- **Tooling:** Customised tooling developed on top of the chosen DDS performance benchmarking tool to enable automated experimentation across an optionally-configured range of QoS settings.
- **Performance Evaluation of DDS**
 - Virtualisation: Insights into how DDS performs in virtualised conditions, including its interaction with MC and malicious attacks.
 - MC: Analysis of how MC impacts DDS performance including counter-intuitive findings based off data collected across all three testbeds with various deployment strategies of participants.
 - Data Length: Insights into how varied data lengths impact DDS in terms of latency and throughput when directly compared as well as its interactions with other QoS parameters.
 - Simulated Network Noise: Analysis into how throttling the BW at varying levels impacts the performance of DDS.

- **ML Application:** Pioneering the application of ML for the prediction of DDS performance as characterised by distribution percentiles. Initial applications involved LR and RF.
- **Performance Evaluation of ML:** Further development of ML modelling with two new models (XGB and FFNNs), data from a larger testbed which better characterises the scalability of DDS, two new levels of simulated network noise, a new approach to the model output, and a deeper analysis into various modelling techniques and data transformations.

The findings and methodologies presented herein are anticipated to benefit researchers, practitioners, and system designers engaged in the development and deployment of DDS.

Chapter 2

Literature Review

2.1 Introduction

DDS is a widely used communication specification that has been used for over two decades. Its usage is embraced by a variety of industries due to its characteristics that facilitate incorporation into real-time, mission-critical, and distributed systems. These industries include, for example, autonomous vehicles, military defence systems, energy management systems, and air traffic management systems. Development of DDS first began in 2001 by Real-Time Innovations (RTI) (*RTI* n.d.) and Thales Group (*Thales Group* n.d.) before the OMG (*Object Management Group* n.d.) published the first version of the DDS specification in 2004.

This chapter serves as the foundational background and literature review for the experimental investigations presented in this thesis. It provides essential context for understanding the design of the experimental campaigns (Chapter 3), the analysis of their results (Chapter 4).

This chapter provides a comprehensive review of the literature on the performance of DDS, focusing on specific implementations. It seeks to answer the question: “How performant is DDS according to various implementations?”. Since the performance of a specification cannot be directly measured, this review considers the performance of various DDS implementations. The chapter focuses on its performance under default QoS settings, and the performance of different DDS implementations as documented in the surveyed literature.

Performance evaluation of DDS is useful for practitioners and researchers as the results demonstrate how different DDS implementations perform under different conditions, and what affects the performance. These results, thus, help understand DDS performance better, and contribute to making decisions that may lead to the usage of DDS over other communication

technologies. More specifically, this thesis investigates the performance of individual DDS implementations that should benefit the decision about which implementation to choose and under what particular conditions. Moreover, these implementations change over time and this can therefore affect the ranking in terms of performance. Furthermore, certain implementations may excel in certain scenarios, e.g. implementation X may have better performance when run in virtualised environments than implementation Y, but the opposite might be true for large data lengths in physical or virtualised environment. A full explanation of DDS and how it works is provided in Section 1.1. At the time of writing there are several available DDS implementations from various companies. The leading DDS products include Connex DDS from RTI (*RTI Connex* n.d.), Vortex OpenSplice from ADLINK (*Vortex OpenSplice* n.d.), FastDDS from eProsima (*eProsima Fast DDS* n.d.), MilDDS from MilSOFT (*MilSOFT* n.d.), and OpenDDS from OCI (*OpenDDS* n.d.). In the surveyed literature RTI product versions ranged from 4.5 to 6, OpenSplice product versions ranged from 5.4.1 to 5.5.1, OpenDDS 3.4.1 was evaluated as well as FastRTPS 1.6.0. Most of these implementations are evaluated within the reviewed papers and their results, and related discussion, will be stated later in this chapter. Despite each implementation following the OMG DDS specification, they differ since the standard does not specify exactly how certain mechanisms must be implemented. In this chapter we attempt to provide a “clean” comparison of DDS implementations by considering equivalent QoS configurations of these products.

The remainder of this chapter is structured as follows: Section 2.2 outlines the systematic literature review methodology used. Section 2.3 presents the findings from the surveyed literature. Section 2.4 discusses the results and identifies key themes. Finally, Section 2.5 summarises the findings and highlights research gaps that this thesis aims to address.

2.2 Research Method

The overall objective of this chapter is to identify what has been researched in the field of experimental evaluation of DDS performance. This includes performance of DDS under default QoS conditions, and specific use cases such as air traffic management, video streaming, etc. To achieve this, we have applied a systematic literature review which is a well-defined and precise method of identifying, evaluating and analysing studies related to the subject of interest. A systematic literature review here aids in identifying any research gaps to create a foundation whereby further investigative research work can be realised. The research method, from which this chapter stems from, is explained in (Kitchenham et al. 2009). This method has been used

throughout other survey papers related to DDS such as (Köksal & Tekinerdogan 2017). There exist six main points of the proposed approach, which have been tailored slightly to suit our survey:

1. Identification of the research question
2. Identification of a search strategy
3. Identification of the study selection criteria
4. Data extraction from studies
5. Data summary from data extractions
6. Critical analysis and theme identifications

The structured process commences with the identification of the research question and any sub-questions. This is in preparation for the following step: identifying the scope, since the questions aid in the creation of the scope for the papers of interest. Since the focus of this chapter is the experimental performance assessment of DDS, the scope is limited to just that and any papers regarding other (related) topics, for example security vulnerabilities of DDS, have been excluded. Once the scope is identified, we can then create a search strategy to collect all (to the best of our knowledge) the papers of interest. This strategy defines where searches will be made and what the search terms are to reproduce these search results if required. After the completion of this step, we are left with a collection of studies from various literature, e.g. journals, conferences and symposiums proceedings, etc. From this initial collection we filter out any unrelated papers where these filtering rules are identified later in Section 2.2.4. This produces a collection from which we extract the data that benefits in answering the given research question. Once the data is extracted and summarised, we critically analyse the literature and identify themes where we group the papers according to their theme. We complete the survey paper with identification of research gaps and possible future work.

2.2.1 Research Question

This thesis focuses on surveying results about experimental performance evaluation of DDS. Since DDS itself is a specification, one cannot meaningfully evaluate its performance and we can instead use a sample of the most reputable DDS implementations to do so. This has led to the research question: “How performant is DDS according to its implementations?” which defines a scope for the rest of this chapter involving two main sub questions.

A research sub-question is as follows: “What is the difference in performance of various DDS implementations?”. Various DDS implementations have been evaluated in the surveyed

literature to determine which implementations excel under which conditions. As mentioned previously within the introduction to Section 2.3.2, a complication is that various DDS implementations have different versions each of which is likely to exhibit a different performance output. We rely on the reviewed literature to provide results for various versions of a given DDS implementation.

From these research sub-questions we generalise on the results to determine how performant the DDS specification is according to the selected implementations.

2.2.2 Scope Identification

Once the research question has been identified, a scope had to be clarified to define what type of papers would be pertinent for this review. We have collected possibly all (to the best of our knowledge and efforts) papers between 2003 and 2025 that are related to DDS and its performance evaluation. The year 2003 has been particularly chosen as this was when the OMG released the DDS specification and 2025 is the time of which this thesis is being written. The scope will also contain all papers that contain performance metrics related to DDS subjects including the application of DDS with other technologies as well as evaluation of the performance of DDS within virtualised environments.

2.2.3 Search Strategy

The search strategy includes any means of identifying potential papers of interest by applying search terms to various literature sources including Google Scholar (*Google Scholar* n.d.), IEEE Xplore, ACM Digital Library, Wiley, Science Direct and Springer. The search terms were carefully chosen to ensure that the papers collected are relevant to the research question and scope and include:

- “data distribution service performance evaluation”
- “data distribution service performance”
- “dds performance”
- “data distribution service performance assessment”
- “dds performance assessment”
- “opensplice dds”
- “rti dds”

A total of 46 pieces of literature were collected including survey papers and PhD theses. The relatively low number of the literature collected points to a lack of research in this area and this number decreases in the next step once the selection criteria are applied.

2.2.4 Selection Criteria

The selection criteria filter out the literature of no interest. We used these criteria to further focus on papers that aid in answering the research question and to remove any papers that are not pertinent. The criteria included accepting papers that are written in English (some papers were written in Korean whilst their abstracts being written in English), accepting papers that included performance test results in the form of graphs or tables, accepting papers that did not change how the DDS specification worked to gain performance improvements (with some exceptions since RTI has implemented features such as AutoThrottle which is an experimental feature that allows the user to configure a Data Writer to automatically adjust its writing rate and send window size to provide the best latency/throughput trade-off while system conditions change as mentioned in section 10.4 of (*RTI Connext DDS Core Libraries User's Manual* 2016)).

After applying the filtering to the 46 papers, the list narrowed down to 29. Out of these, 17 papers were published in conference proceedings, 10 in journals and 2 had unidentifiable sources (originally found as PDF (Portable Document Format) files resulting from a Google Scholar search).

2.2.5 Data Extraction

After collecting a considerable number of papers, we extracted information related to the research question and we had to identify the common data that occurred frequently throughout all papers. This would include items of interest to the performance evaluation of DDS implementations as listed below:

- Data Length of DDS Messages
- Parameters Investigated During the Test (including both QoS and non-QoS parameters)
- Number of Publishers
- Number of Subscribers
- Products Evaluated
 - Product Version
- Publication

- Publication Type
- Paper Subject
- Number of Experimented Parameters
- Number of Products Evaluated

For each paper, we identified exactly what performance measurements were taken and then recorded the conditions of the tests under which the measurements were obtained. This was followed by the summary of the tests along with its corresponding conditions such as QoS values and various items associated with its description. The data lengths recorded are the sizes of the DDS messages sent and received during the tests whilst the number of publishers and subscribers are self-explanatory: they are the respective participant amounts present in a given test. The parameters investigated during the tests include both QoS and non-QoS-related parameters where the parameters' values were explored and experimented with. Table 2.1 lists the parameters gathered with their corresponding descriptions.

Parameter	Description
AutoThrottle	Automatically adjusts the writing rate and the send window size.
Batching	Batches samples together to send instead of sending one sample at a time.
Deadline	Amount of time either between writing each sample on a DataWriter or reading each sample on a DataReader.
Destination Order	Controls what sample is chosen when multiple DataWriters send the same sample to a DataReader.
Durability	Controls whether data reader that join late can access already published data.
History	Controls how many samples to be stored on either the DataWriter or DataReader.
Latency Budget	Provides a hint to the maximum acceptable delay from the time the data is written to the time it is received by the subscribing applications.
Life Span	Controls how long a sample can be stored in cache before being deleted.
Liveliness	Controls whether a DataWriter is “alive” in order to be used by the ownership parameter.
Maximum Transmission Unit (MTU)	Size of the largest packet size that can be sent on the network.
Message Size	Data length, size of the DDS message.
MC	Controls whether the data can be sent one at a time or to multiple nodes at once.
Number of Nodes	Amount of nodes on the network containing publishers and subscribers.
Presentation	Controls how the data arrives at the data reader.
Publisher Amount	Number of publishers used in the test.
Publishing Rate	Samples published per second.
Reliability	Controls whether missed samples are resent or not.
Subscriber Amount	Number of subscribers used in the test.
Transport Priority	Optional QoS parameter that describes the priority of the sample being sent.
TurboMode	Adjusts the number of B in a batch at runtime according to current system conditions.

Table 2.1: Parameters and their descriptions gathered for all papers in the literature review on the experimental performance evaluation of DDS.

DDS implementations that were evaluated based on the surveyed papers are as follows: Vortex OpenSplice, RTI's DDS, FastDDS, and OpenDDS. The DDS products allow users to create required DDS applications and should not be confused with associated benchmarking tools, such as PerfTest (*RTI PerfTest* n.d.) provided by RTI to evaluate and monitor DDS performance (of their, and other, products), via assessing latency, throughput, CPU utilisation, etc.

2.2.6 Data Summary

Categorisation of the papers began by classifying the general themes of the papers followed by documentation of the category. This led to some categories containing only a small number of, or just one single, piece of literature indicating that these were too specific. Therefore, once all the papers were surveyed and all categories were identified, we grouped the papers into the following more generalised categories:

- Performance of DDS
- Performance of Applied DDS
- Performance of Virtualised DDS

Performance of DDS This is the main category of interest for this chapter. These papers evaluated the performance of DDS, including compared multiple DDS implementations and experimented how specific different QoS and non-QoS parameters affected the performance of DDS.

Performance of Applied DDS Authors who applied DDS to certain scenarios or integrated it with other technologies will have their papers placed under this category. This differs from the previous type as with the previous types various test conditions were chosen depending on what values were being investigated and how these values affected the performance. On the other hand, this category specifically looked at test conditions that were customised for certain scenarios whereby the test parameters were carefully considered to produce an optimal performance result.

Performance of Virtualised DDS This category contains all papers that evaluated the performance of DDS when run in virtualised environments.

2.2.7 Critical Analysis and Theme Identification

The following themes were identified from the literature survey:

- Evaluation of Multiple DDS Implementations
- Evaluation of a Single DDS Implementation
- Evaluation of Unnamed DDS Implementations
- Evaluation of DDS Implementations in a Virtualised Environment

Some papers belong to more than one theme as they contained multiple types of tests. In these cases, we have extracted the parts that correspond to a given theme and included it in the related section. Thus a paper’s results can be found in more than one section.

Throughout the extraction of the test results, we have focused on three main performance metrics: *latency*, *throughput*, and *jitter*. This has mainly stemmed from the fact that these were the main metrics provided by majority of the papers. This was to be expected since these three metrics are vital in evaluating the performance of any publish-subscribe technology. The latency is usually measured in ms though sometimes when values are small enough, the microseconds (μs) units are used. It should also be stated that some papers measured the latency using the system clocks where both publisher(s) and subscriber(s) resided, and used certain techniques to increase synchronicity between the timings. On the other hand, other papers measured the latency on the publisher side only, by taking first the measurement a data sample is published by a given publisher, and subsequently when the acknowledgement that a subscriber has obtained the sample is received by the publisher. In this way the two timestamps are taken on the same system where the publisher resides. This value, known as the Round Trip Time (RTT) would then be halved resulting in a one-way latency. In either way, the true one-way latency is not really obtained since the value cannot exactly be attained due to lack of true synchronicity or the fact that the second “half” of the two-way measurement could affect the value of the actual one-way latency and this therefore means that a “fair” comparison cannot exactly be made which reinforces the gap in the research about the accuracy of the latency measurements. However, there are well known approaches for attempting to get as close as possible to this true synchronicity such as (Lamport 1978).

2.3 Results

In this section we have summarised all the results from the surveyed literature and it should be noted that when a paper mentions that the enclosed tests are characterised with the “default”

QoS settings, it refers to the default values stated in the DDS specification.

2.3.1 Overview of Selected Studies

A total of 29 papers were thoroughly reviewed throughout this section. From these papers, the number of publishers involved within the reviewed experiments ranged from 1 to 45 whilst the subscriber amount ranged from 1 to 100. Furthermore, 7 papers compared the performance of numerous versions of various DDS implementations, 11 papers evaluated the performance of specific DDS implementations where the tests focused on certain experimental conditions, 4 papers evaluated the performance of DDS implementations without mentioning their names or versions, and finally, 6 papers evaluated the performance of DDS when executed in a virtualised environment though 3 of these papers had similar authors and content. The test with the most widely ranged configuration encompassed 8 different test variables (including QoS parameters) and the most popular parameter was the message size which was evaluated in 13 of the papers followed by the reliability parameter being evaluated in 8 papers. Connex by RTI was the most popular vendor evaluated with 19 papers investigating its performance whilst FastRTPS was the least popular product with only 4 papers.

2.3.2 Evaluation of Multiple Data Distribution Service Implementations

A total of six pieces of literature have been included within this section and all these papers have evaluated at least two different implementations of DDS. Since these papers have been published at different times the names of these DDS implementations may have changed and therefore, to ensure consistency throughout the results, we refer to the ADLINK implementation as OpenSplice, the OCI implementation as OpenDDS and finally the RTI implementation as RTI DDS (though in many cases it was referred to by its most recent title: Connex).

Data Length

All 6 papers evaluated the performance of DDS when varying the values of data length ranging from 2 B to 4 megabyte (MB). Typical UDP packets are 64 KB in size and therefore any messages bigger than this are split into multiple packets and observing how this affects the performance would be interesting since theoretically, large data lengths should increase the latency and throughput.

The piece of literature (Krinkin et al. 2018) tested data lengths ranging from 100 B to 100 KB and found that OpenSplice was slower than OpenDDS in terms of latency and jitter (lower latency and higher jitter). This paper also compares the performance of OpenSplice, OpenDDS and FastRTPS with data lengths ranging from 100 B to 1 MB where the results from this comparison show us that OpenSplice has the highest latency, followed closely by FastRTPS’s latency, and OpenDDS had the lowest latency of the three implementations whilst latency increases linearly as the time progresses for all implementations. In summary, this result shows us that whilst the latency linearly increases for all implementations (as the data length increases), OpenSplice has the highest latency, followed by FastRTPS and OpenDDS with the lowest latency. It must be noted that the paper mentions that all the implementations were tested in “safe” mode, meaning that all messages were stored in memory the entire time as well as the fact that every message was followed with a confirmation about its delivery and this may be the reason why the processing time of each next message went up.

Performance evaluation of OpenSplice and RTI DDS was carried out in (Bellavista et al. 2013) whilst varying the data length between 128 B and 512 KB. The results in Figure 4b of the mentioned paper shows that OpenSplice has a higher throughput up to the 4 KB mark. From that point onwards RTI DDS significantly outperforms OpenSplice in terms of throughput since RTI DDS has a self-contained decentralised architecture. OpenSplice crashes once a data length of 112 KB is experimented on while RTI DDS continues performing and the authors explain that this could be because of the saturation of the DCPS daemon continuing to consume RAM and CPU until the crash due to such a heavy system overload. It is also stated that after 64 KB, the message must be split into multiple UDP packets which is the reason why the throughput remains the same while the rate decreases. Data is being transported and DDS is waiting for data to be sent before it starts writing again and RTI DDS’s defragmentation seems to be highly optimised as mentioned by the authors of the paper. The paper also mentions that RTI DDS saturates the bandwidth when a data length of 16 KB was used. In summary, these results show that OpenSplice outperforms RTI DDS up to data lengths of 4 KB and for data lengths greater, RTI DDS outperforms OpenSplice in terms of throughput.

Performance of OpenSplice for data lengths ranging from 256 B to 4 MB amongst many other QoS configurations was carried out in (Maruyama et al. 2016). Figure 7 of the mentioned paper shows that the latency of OpenSplice is constant for data lengths up to 4 KB where it begins increasing non-linearly to the point where it reaches around 6 ms for data lengths of 64 KB. Meanwhile, Figure 9 of the mentioned paper surprisingly shows a different outcome: despite the increase in data length, the latency remains constant. This is because DDS requires marshalling

various configurations and decisions for the QoS policy. The authors mention that they observe a trade-off between latencies and QoS policy regardless of the data size as well as the fact that the latencies are predictable and small even though the QoS configurations produced overhead. Figure 9 shows us that the data length does not affect OpenSplice’s latency up to 64 KB whilst Figure 10 of the mentioned paper shows a non-linear increase up to 4 MB before a sharp increase in the latency when large data is split into several datagrams. From these results we can summarise that OpenSplice’s latency increases while the data length increases as with all DDS implementations and the degree of the increase is determined by whether the nodes are local or remote. Specifically, OpenSplice seems to handle small data lengths significantly better than larger data lengths.

Evaluation and comparison of the performance of OpenSplice and RTI DDS was carried out in (Fujdiak et al. 2019) for data lengths ranging from 100 KB to 1 MB and the results seen in Figure 6 of the mentioned paper show us that OpenSplice’s latency is significantly greater than its competitors RTI DDS and OpenDDS which both have similar latency values. From these results we can deduce that OpenSplice is not well-suited for large data lengths, a solidifying trend.

It has been clearly shown that for larger data lengths, OpenSplice is outperformed by its competitors. However, the question in focus now is how does it perform in comparison for smaller data lengths and are the differences significant? We know from (Bellavista et al. 2013) and (Maruyama et al. 2016) that OpenSplice’s performance is underwhelming for data lengths greater than 4 KB, so let us investigate the comparison of OpenSplice’s performance for data lengths below 4 KB.

OpenSplice outperforms RTI DDS in (Bellavista et al. 2013) in terms of throughput for data lengths between 128 B and 512 KB mainly because of batching. Figure 9 of (Maruyama et al. 2016) shows us that OpenSplice outperforms RTI DDS with a lower latency for all data lengths from 256 B to 4 KB though it should be noted that OpenSplice is using the “reliable” reliability setting while RTI DDS is using the “best effort” reliability setting in this graph which is an example of an “unfair” comparison. Therefore, from these results we can summarise that OpenSplice outperforms RTI DDS in terms of throughput for data lengths smaller than 4 KB.

At this point, OpenSplice outperforms RTI DDS for smaller data lengths and is outperformed by all implementations for larger data lengths. In the following paragraphs we will investigate how the other implementations compare with each other when the data lengths vary.

Figure 4b of (Krinkin et al. 2018) shows us that for data lengths ranging from 100 B to

10 KB, OpenDDS outperforms FastRTPS with a lower latency. Meanwhile, for data lengths ranging from 10 KB to 1 MB, the opposite is the case; FastRTPS outperforms OpenDDS with a lower latency. Therefore, it seems that OpenDDS is also better suited for smaller data lengths up to 10 KB whilst FastRTPS is better suited for larger data lengths ranging from 10 KB to 1 MB.

Figure 6 of (Fujdiak et al. 2019) shows us that between 100 to 500 KB and from 700 to 1000 KB, RTI DDS subtly outperforms OpenDDS with slightly lower delays. A strange result is shown between 500 and 700 KB since OpenDDS outperforms RTI DDS. Comparison of the performance of RTI DDS and OpenDDS was done in (Esposito et al. 2008) and shows us that between 32 B and 8 KB, RTI DDS has a lower latency though the difference is not significant, and the jitter tells us that RTI DDS has more stable behaviour than OpenDDS. For data lengths greater than 8 KB, OpenDDS outperforms RTI DDS where the difference in performance increases with the data length. Meanwhile, the authors in (Profanter et al. 2019) evaluated OpenDDS and FastRTPS for data lengths ranging from 2 B to 32 KB and mentioned that OpenDDS is considerably slower than FastRTPS throughout the tests.

In conclusion, the literature has shown that for smaller data lengths ranging from 1 B to 4 KB, OpenSplice is very performant as well as OpenDDS (though up to 10 KB in this case). The performance of these implementations substantially worsens as the data lengths increase. For larger data lengths ranging from 10 KB onwards, RTI DDS and FastRTPS are more performant and better suited.

Reliability

Theoretically, the reliable value of the setting has mechanisms that introduce an overhead which introduces the question: How significant is this overhead compared to best effort and how does it impact the latency especially and what implementations perform best when using each value?

The first paper, (Bellavista et al. 2013), ran two types of tests that although had different reliability values, also had different values for other QoS settings. The first test experimented with one million packets and best effort whilst the second test looked at 5000 packets and reliable communication. The difference in the number of packets significantly affects the results and does not explicitly show us the impact of the reliability parameter since the authors intended to measure the throughput and samples per second for the first type of test whilst focusing on the latency for the second type of test. The results state that the RTT of the test where 5000 packets were sent using the reliable value was lower than the test where one million packets

were sent using best effort. This is to be expected but exactly how different was the round-trip time? We cannot answer this question as the authors of the paper do not share the exact measurements or the RTT of the first test (one million packets using best effort) as they focused on the throughput metrics as mentioned.

Another paper, (Maruyama et al. 2016), did a better job at identifying the impact of best effort versus reliable although multiple QoS values were also set during the experiments (confounding the interpretation of the results on how best effort affects the performance of DDS). The paper experimented on RTI DDS and OpenSplice in a remote as well as a local environment. The results shown in Table 4 of the mentioned paper explain that it is impossible for RTI DDS to use reliable data greater than 128 KB whilst OpenSplice can. The authors mention that this is due to a deficiency of additional configurations for large data on RTI DDS. These results were obtained when running the DDS implementations remotely. Furthermore, when ran remotely with best effort the results for both RTI DDS and OpenSplice were the same. Data transport was possible for all data lengths except for 2 and 4 MB where the data transport was still possible but was missing the deadline.

When running the tests locally (where messages were passed over a local loopback), it was again impossible for RTI DDS to use data greater than 128 KB whilst OpenSplice was able to achieve this with the same reason mentioned before. The rest of the results were the same as mentioned before except when 4 MB of data being sent on RTI DDS using best effort was impossible due to a halt of process or too much data loss. Figure 7 of the mentioned paper compares the latencies of OpenSplice using reliable and RTI DDS using best effort communication and the results are quite intriguing as the graph demonstrates that the latency of RTI DDS while using best effort communication is in fact larger than when using OpenSplice with reliable communication meaning that the best effort method took longer. The difference increases as the data lengths increase though we cannot exactly deduce that best effort produces a larger latency as different DDS implementations were used and we cannot cleanly compare the results as they tend to implement the reliability mechanisms in different ways. This trend of RTI DDS best effort having a higher latency than OpenSplice reliable is shown throughout the paper in figures 8, 9 and 10 as well.

Number of Subscribers

Overall, 2 papers evaluated the performance of DDS when varying the number of subscribers ranging from 1 subscriber to 45. The number of subscribers that are present can affect the

performance of DDS to a varying degree depending on other QoS parameters. On top of this, we intend to find out what DDS implementations excel in terms of scalability. It would be useful to first identify the impact of the number of subscribers with the default QoS settings and then to experiment with various QoS values alongside changing the subscriber amount. However, in terms of what has been done already, (Krinkin et al. 2018) experiments with 1, 9 and 45 subscribers in 3 separate tests. In the first test with 1 publisher and 1 subscriber it is seen in Figure 4b of the mentioned paper that OpenSplice takes the longest to complete the experiments throughout. Meanwhile, it is quite clear that FastRTPS has the second longest duration followed by OpenDDS with the fastest time up to 10 KB. When compared with the second test where 9 publishers and subscribers are present, OpenSplice still takes the longest to complete the tests throughout all data lengths while OpenDDS and FastRTPS seem to have similar measurements in terms of how long the tests took up to 10 KB. Between 10 KB and 1 MB, FastRTPS took longer than OpenDDS to send the packets. In the final test where 45 publishers and subscribers are present, it is again shown that OpenSplice take the longest. However, throughout the different data lengths, both FastRTPS and OpenDDS have similar durations and from these results we can see that OpenSplice does not handle scalability well and the authors of the mentioned paper also state that OpenSplice has no ability to run a lot of participants as it has a limit of 120 participants. From this we can determine that OpenSplice is not designed for larger systems with numerous participants and large data lengths.

Scalability focused tests were ran in (Esposito et al. 2008) comparing RTI DDS and OpenDDS where they experimented with 1, 2, 4, and 7 subscribers. For 1 and 2 subscribers, both RTI DDS and OpenDDS had almost the exact same one-way transit time whilst for 4 and 7 subscribers, RTI DDS outperforms OpenDDS in terms of latency. From these results we can determine that RTI DDS handles scalability better than OpenDDS for 4 or more subscribers up to 7 subscribers.

2.3.3 Evaluation of a Single Data Distribution Service Implementation

The papers mentioned within this section have evaluated a single DDS implementation to either evaluate the performance of DDS within a certain scenario, compare novel solutions with traditional DDS solutions in terms of performance or even applying DDS to alternative standards and then evaluating its performance. The usefulness of this specific segment of the literature review may result in a reinforcement of what has been previously stated in Section

2.3.2 or the results attained may even be significantly different and will therefore produce new questions.

Evaluation of DDS using Default QoS Settings

DDS has many competitors in terms of network communication standards and evaluating the performance of a system using the default QoS settings as stated by the OMG standard provides an adequate foundation for identifying the base performance of DDS. One can then tune these parameters to identify performance benefits or detriments and we intend to discover what has been identified in terms of the base performance of DDS using the default settings. The exact default parameter values are listed in Table 2.2.

Parameter	Default Value
Durability	volatile
Deadline	infinite
Latency Budget	0
Reliability	reliable
History	keep_last

Table 2.2: Default QoS parameter values.

There are other parameters where the mentioned papers have evaluated multiple values such as data length, total messages sent, total duration of the test, number of publishers and subscribers, and the locations where these publishers and subscribers reside, i.e., how many machines exist and what participants they accommodate for. With this in mind, we can analyse the performance of DDS according to various data lengths before identifying how the performance is affected over time throughout the duration of the test.

When a subscriber and a publisher are ran on the same local machine with the default QoS settings and 500 B messages are disseminated, (Essers & Vaneker 2014) shows us in Figure 6 of the paper that the highest latency attained is 50 ms with an average latency of around 25 ms for the OpenDDS implementation. Meanwhile, (Almadani 2016) looked at implementing DDS within the petroleum industry and ran tests using the default QoS settings as mentioned using the RTI DDS implementation and the results when compared to the previous paper are similar. For 512 B messages, the results show an average latency of 18 ms which is lower than that of OpenDDS (25 ms).

When varying the data lengths, Figure 8 of the mentioned paper shows that the latency increases exponentially after being constant up to 256 B and this trend was present when experimenting with one publisher and multiple subscribers. This pattern was also seen in DDS

for industrial automation where the latency was constant up to 512 B before exponentially increasing.

To summarise, on average, latencies seem to be constant up to 256 B before exponentially increasing while for message sizes of 512 B, an average latency between 18 to 25 ms is achieved.

Performance Impact of Reliability

One of the most popular experiments carried out throughout the analysed literature includes the comparison of performance when the reliability is set to best effort versus reliable. Theoretically, the best effort reliability should provide a lower latency as there are no mechanisms that introduce overheads confirming message dissemination. However, in this section we aim to investigate how significant the influence of the introduced performance overhead is and therefore will become a vital deciding factor that may lead to the usage of a reliable setting. This section is different to Section 2.3.2 since we are not comparing the performance of two different DDS implementations but rather the performance of the same DDS implementation with different reliability values.

Figure 6 of (Do & Franklin 2011) shows us that reliable and best effort values of the same QoS setting combinations seems to have similar results in terms of latency when ran on the RTI DDS implementation.

Figure 10 of (Yang et al. 2012) indicates that for data lengths between 16 B to 128 B, reliable has a higher latency as expected. For messages of 8 B, it seems that the latency produced from both best effort and reliable seem to be almost the same and this also seems to be the case for 256 B messages as well as 4096 B messages. In terms of throughput, (Almadani 2016) shows us in Figure 11 that for 2 and 4 subscribers, both reliable and best effort produce the same throughput. For any more subscribers, the reliable reliability produces more throughput as expected.

To summarise, theoretically we expect the latency and throughput of reliable tests to be higher than that of best effort. This has proven to be true in most cases, though for messages of 8, 256, and 4096 B the latency seems to be almost the same for both reliable and best effort. Furthermore, when up to 4 subscribers are present, the effects of the reliable value are not shown.

Performance Impact of Quality of Service Tuning

Two papers have looked at tuning the QoS settings to suit their scenarios to gain the best performance. We can therefore collect the results from these results to determine how the changes in QoS affected the performance.

The first paper, (Almadani, Aliyu & Shakshuki 2015) ran tests with a durability of “volatile”, reliability of “reliable”, liveliness of “automatic”, history of “keep_last” with a depth of 5 (instead of 1), owner of “exclusive” (instead of shared), owner strength of 5 (instead of 0), and a minimum separation value of 0.2 seconds (instead of 0) whilst using RTI DDS. A total of 100,000 messages of 28 B in length were sent and the performance both with and without the QoS settings were compared. In terms of throughput, the results were very similar, and the differences were not noticeable. The tuned QoS tests had an average throughput of 769 Mbps whilst the default settings produced an average throughput of 778 Mbps. When looking at the latency results, of the 10 iterations, 5 produced lower latencies with the tuned QoS settings whilst the other 5 produced higher latencies. It is worth noting that on the 6th and 7th iteration, the tuned QoS tests produced the highest latencies of 5500 to 7000 μ s whilst the default QoS parameters had latencies of 3500 and 3200 μ s respectively. Figure 7 of the mentioned paper shows that these experimental results seem to be outliers.

Another paper, (Al-Madani & Ali 2017), evaluated the performance using a durability of “transient_local”, a reliability of “best effort” on the publisher and “reliable” on the subscriber, a history of “keep_all” on the publisher and “keep_last” on the subscriber, and unlimited resource limit on the publisher and a value of 1 for the subscriber. The tests also compared between Local Area Network (LAN) and Wide Area Network (WAN) and increased the number of publishers and subscribers up to 6. The results for the default QoS settings show a lower latency throughout all publisher and subscriber amounts. There exists a difference of around 5 ms between the default and non-default QoS settings with the default QoS producing lower latencies which is an outrageous difference. Overall, the latency linearly increases as the participants increase and these results are the same for the WAN tests. The throughput follows the same pattern though the default QoS settings has less throughput in both the LAN and WAN cases.

Performance Impact of Scalability

These reviewed papers have investigated the effect on the performance of DDS in terms of scalability which includes varying the number of publishers, subscribers, DataWriters, and DataReaders.

Throughout the reviewed literature, two papers have investigated the effects of adding participants. The first, (Rhoades et al. 2007) experimented with values of data lengths ranging from 128 to 32768 B and increased the number of subscribers up to 7 whilst using a true MC setting. The results show that the median latencies increase linearly as the number of subscribers increase for data lengths up to 8192 B. Furthermore, the increase for 32768 B messages in terms of latency from 1 subscriber to 4 was greater than from 4 subscribers to 7 and it must also be noted that the 32768 B messages had outrageous maximum latencies of around 5000 μ s. Moreover, when 2 subscribers were used, the 128 B messages had a higher maximum latency than all other lengths up to 8192 B. As well as this, 4 KB messages with 7 subscribers had a maximum latency of 2000 μ s while 8 KB messages had a maximum latency of just over 1000 μ s.

The second paper, (Almadani, Bajwa, Yang & Saif 2015) ran tests using messages of 1024 B, 1 publisher and various subscriber amounts. For the experiments where 1 publisher is used, the latency increases linearly with the number of subscribers. When 2 to 4 publishers were active the latency did not follow the same pattern as before. In fact, the latency decreased when a second subscriber was added and then continued to increase for all other additional subscribers. In terms of the throughput, 150 B messages were transported and in the case of 1 publisher and many subscribers, the throughput seemed to remain constant throughout the increase of subscribers with a slight decrease from 6 subscribers to 8. Figure 9 of the mentioned paper shows that when multiple publishers are used the increase in publishers seems to increase the throughput when comparing 2 publishers to 4. It seems an increase in subscribers does not affect the throughput.

Other Experimental Findings

Other experimental findings are those that have not exactly been repeated in multiple papers. These tend to be papers that have carefully tuned the QoS parameters to suit specific scenarios that the authors were interested in.

Performance of DDS when both on LAN and WAN was carried out in (Do & Franklin 2011). The paper alternates between using a 100 Mbps LAN ethernet versus a 54 Mbps WAN connection. The results show that for messages of 28 B in length, a minimum latency of 375 μ s is attained on the LAN connection. Furthermore, the paper mentions that the WAN experimental results tend to have high variance whilst the LAN experimental results were much more stable. From these results we can conclude that LAN tests are preferred due to its stability though

WAN is capable of being used when required.

Evaluation of the performance of DDS when varying values of the latency budget was carried out in (Yang et al. 2012). The authors experimented with values between 0 and 100 and the results of the experiments found that a higher latency budget led to a higher average latency with only a difference of 40 μ s. The case was the opposite for the maximum latency values. It was also found that a higher latency budget led to higher jitter.

Overall, we have seen from the non-categorised experiments on individual DDS implementations that using MC improves application performance with multiple participants, WAN measurements have high variance compared to LAN, a higher latency budget leads to a higher average latency, lower maximum latency, and a higher jitter, TurboMode improves throughput, and finally, AutoThrottle reduces the latency at the cost of throughput.

2.3.4 Evaluation of Unnamed Data Distribution Service Implementations

Four pieces of literature evaluated the performance of DDS and the authors had not specifically stated the implementation that they used or may have used simulations.

Evaluation of Scalability on DDS using Simulations

Two papers investigated the effects of DDS under network simulations whilst increasing the number of participants for each test. It should be stated that simulation performance may not be the same as the performance from a physical setup and should therefore not be compared. The piece of literature (Lee et al. 2015) ran tests with 1024 B messages whilst increasing the number of subscribers from 20 to 100. The results show that the latency increases linearly with the number of subscribers with the increase between 20 and 60 subscribers being greater than the increase between 60 and 100 subscribers. The results also show that the latency ranges from 5 ms at 20 subscribers to 7 ms at 100 subscribers. There is an almost linear increase in the number of messages in correspondence with the increase in subscribers. At 20 subscribers there are around 1000 messages being sent whilst at 100 subscribers there are around 13,500 messages being sent which is a much larger increase than what was seen in the latency.

Another paper, (Kim et al. 2015) evaluated the performance of DDS when varying the number of DataWriters and DataReaders over 50 and 100 participants. Eight scenarios were run increasing the number of DataWriters and DataReaders by 50 per scenario resulting in the first scenario having the least number of DataWriters and DataReaders and the eighth scenario

having the greatest number of DataWriters and DataReaders. The results show a strange pattern. Scenario 5, with 100 participants, 200 DataWriters and 200 DataReaders had the lowest throughput in terms of messages per participant whilst Scenario 4 with 50 participants, 350 DataWriters and 350 DataReaders had the highest throughput. A linear increase is then seen from Scenario 5 up to Scenario 8. It seems that the throughput shown in terms of messages per participant increases with the amount of DataWriters and DataReaders.

In summary, 2 papers have evaluated the performance of DDS whilst increasing the number of participants or DataWriters/DataReaders. Generally, the latency increases with the number of subscribers and so does the throughput though much more significantly. When adding DataWriters and DataReaders a strange pattern is observed where 100 participants with 200 DataWriters and DataReaders produces a lower throughput than 50 participants with 350 DataWriters and DataReaders though the throughput generally increases with the number of DataWriters/DataReaders.

Evaluation of the Impact of Multicast

Two papers also investigated the impact of the use of the MC setting. Both papers ran tests with the exact same configurations (though one varied the messaging rate from 50 to 100 samples per second). Both papers sent a total of 10000 messages using the best effort reliability and varied the message sizes from 32 B to 63 KB.

Focusing on the UC results, (Youssef et al. 2015) showed a very shallow linear increase up to 8192 B before an exponential rise from 8192 B to 63 KB in terms of latency. The second paper, (Youssef et al. 2018) also shows a linear increase in the latency from 32 to 8192 B with an exponential rise from 8192 B to 63 KB.

The first paper mentioned found similar patterns as the UC results within its MC results with a higher latency at 63 KB. The second paper shows that the use of MC provides a constant latency up to 512 B with a linear increase from 512 to 8192 B followed by an exponential rise from 8192 to 63 KB which is very similar to the UC results. In terms of the actual latency values being impacted by MC, the results show that using MC seems to increase the latency very slightly overall for all data lengths in both papers with the second paper showing a larger increase.

2.3.5 Evaluation of Data Distribution Service Implementations in a Virtualised Environment

This section will look at papers that have evaluated the performance of DDS whilst using virtual machines to deploy the DDS implementations. A total of five papers have been categorised under this section though three of these papers were written by Marisol Garcia-Valls, Rosbel Serrano-Torres, and Pablo Basanta-Val and include the same content. These three papers are (García-Valls & Basanta-Val 2017), (Basanta-val & Serrano-torres n.d.) and (Serrano-Torres et al. 2013).

These papers ran tests with default QoS parameters, 1 publisher, and 1 subscriber and the focus of these papers was to identify the effects of virtualisation on DDS. In terms of virtualisation overhead, the results dictate that small data has more virtualisation overhead and RTI has more virtualisation overhead than OpenSplice for both small and medium data. When looking at the results in terms of blocking time, its seen that RTI has a greater blocking time in general than OpenSplice though virtualisation significantly increases the blocking time for both implementations.

Overall, it seems that OpenSplice is better suited for virtualisation than RTI DDS from these results since RTI has less invocations per second, more virtualisation overhead, and a greater blocking time.

Tests were run on virtual machines in (Ferreira & De Oliveira 2017). A total of 11000 messages of 250 B were sent during the test along with MC. The results in the graphs of the paper show an average latency in the range between 270 and 320 μ s. The paper states that the maximum latencies vary from 1 to 5 ms but are less than one percent of the total samples and the quantile graphic shows that more than 95% of the samples are very close to the median.

2.4 Discussion

2.4.1 General Discussion

This chapter has analysed the literature and identified the results for several categories; evaluation of single, multiple, and unnamed DDS implementations, and evaluation of DDS within a virtualised environment. In this section we intend to generalise on the findings from the previous section and to identify any points of interest. It is vital to state that any products that are described as having a better performance is only a result of the survey of the mentioned literature throughout the paper and does not explicitly mean that the certain product

will always behave in the described way.

As mentioned before, to compare multiple DDS implementations, it's best to narrow the results down according to each configuration parameter. Starting with the data length, the results show us that OpenSplice and OpenDDS are better suited for small data whilst RTI DDS and FastRTPS are better suited for larger data. When looking at reliability, the literature has shown that OpenSplice can use reliable data over 128 KB messages whilst RTI DDS cannot. On top of this, RTI DDS seems to lack the ability to send 4 MB messages using best effort and it's also shown that RTI DDS's latency when using best effort is greater than OpenSplice's latency when using reliable though it's not recommended to compare these results as they are different implementations with different reliability settings. When looking at scalability and increasing the number of subscribers, OpenSplice is not well-suited for scalability taking the longest to complete all tests. OpenDDS and FastRTPS generally have similar results and are faster than OpenSplice. RTI DDS outperforms OpenDDS in terms of latency for subscribers ranging from 4 to 7. Therefore, it seems that RTI DDS may be the most performant when focusing on scalability though an inadequate amount of experiments have been carried out to solidify this hypothesis as well as the fact that it is quite difficult to justify this increase in scalability as the gap is minimal jumping from 4 to 7 subscribers compared to a significant increase in the tens or hundreds.

We can use the results from the papers that evaluated a single DDS implementation to solidify what has been presented for certain implementations and to also see how various QoS settings affect the performance of DDS when their default values are used. For these default settings, it seems that RTI DDS has a lower average latency than OpenDDS. From these results we cannot confidently say that RTI DDS outperforms OpenDDS though from what is seen, it does for default QoS settings. Looking at the effects of reliability on the performance of DDS is unclear. The literature has shown that reliable and best effort seem to produce similar results as well as the fact that the reliable value increases the throughput for more than 4 subscribers. Finally, when focusing on scalability, it's strange that the increase in latency from one subscriber to four was greater than the increase in latency from four subscribers to seven. This could mean that as more subscribers are added, the latency increase is reduced indicating a non-linear trend. The results also show that when 7 subscribers are used, 4 KB messages had a higher maximum latency than 8 KB messages. Furthermore, the results indicate that increasing the number of subscribers does not affect the throughput which is opposite to our theoretical understanding. Increasing the number of subscribers ensures that more data must be sent which should therefore lead to an increase in the throughput. However, we cannot

state that this reason is concrete as this depends on how loaded the network is as well as other factors.

Again, we can find the results useful when evaluating unnamed DDS implementations to identify how DDS performs and when looking at the scalability of DDS under simulations its seen that the latency only increases by 2 ms when the subscribers increase by 80 whilst the number of messages sent increases by 12500. This indicates that the throughput drastically increases which is the opposite of what was stated/seen in the previous paragraph. MC does not seem to significantly impact the performance of DDS. Overall, adding MC seems to increase the latency instead of decreasing it, though this statement is only proven by two papers and should therefore be taken lightly. Finally, in terms of the performance of DDS under virtualised environments, OpenSplice is better suited for being run in a virtualised environment than RTI DDS as OpenSplice in a virtualised environment has more invocations per second, less overhead, and less blocking time even though for both implementations, virtualisation takes up a significant portion of the total execution time.

2.5 Conclusion

DDS has been used for almost three decades and its applications are numerous ranging from air traffic control systems to autonomous vehicles. Over the years DDS has been improved and various features have been added to it despite it already being flexible due to the ability to control various QoS parameters and hopefully DDS continues to improve in the future. Investigating how DDS performs under what conditions can help us understand where its weaknesses and strengths lie and what one can do to improve this specification further. However, the literature evaluating the performance of DDS has been quite minimal and this has been identified within this chapter. As well as this, majority of the papers do not look at distribution of measurements and are instead based on the point estimates which indicates a significant gap in the research.

In this chapter, we have carried out a systematic literature review on the evaluation of the performance of DDS. From the many papers on DDS we identified 46 that mentioned performance of DDS and may or may not have included performance metrics. Applying our filtering, we narrowed these papers down to 29 from which we analysed the content and generalised on the themes and findings. The categories explored include evaluation of multiple DDS implementations, evaluation of a single DDS implementation, evaluation of unnamed DDS implementations, and evaluation of DDS implementations in a virtualised environment. We have extracted the various experimental performance tests carried out on DDS as well as the results

from these experiments and have identified certain themes within the research as well as specific gaps.

This chapter has only reviewed the performance of DDS under experimental evaluations and we believe that there is still more to be done in terms of evaluating the performance of DDS, such as model-based evaluations of DDS, experimental evaluations of security-enabled DDS, and model-based evaluations of security-enabled DDS. A literature review on the model-based prediction of DDS performance is contained within Chapter 5.

Chapter 3

Experimentation

3.1 Introduction

The performance of DDS is critical for the real-time systems that depend on it, yet its complexity makes performance evaluation a significant challenge. This difficulty stems from a vast configuration space, with dozens of interacting QoS parameters, and a high sensitivity to the underlying hardware and network infrastructure. Predicting how a specific configuration will behave under certain conditions is notoriously difficult without empirical data. This chapter details the rigorous experimental methodology designed and executed to generate many comprehensive, foundational datasets for the purpose of understanding and predicting DDS performance. The work described herein serves as the bedrock for the subsequent performance analysis in Chapter 4 and the predictive modelling in Chapter 5.

To navigate this complexity, we adopted a systematic, bottom-up research methodology. We begin by detailing the design and rationale for our experimental environments. This includes two distinct physical testbeds and a virtualised platform, chosen to ensure the validity, robustness, and reproducibility of our findings across different infrastructures. We then introduce the custom software suite developed for this research: a benchmarking tool for performance measurement (RTI's Perftest), an automation framework to manage the large-scale execution of thousands of unique configurations, and an analysis tool to explore the impact of DDS QoS parameters.

Following the description of the infrastructure and tooling, we describe the series of experimental campaigns that were conducted. Each campaign was designed to systematically investigate specific hypotheses regarding the influence of factors like message size, network noise, and communication patterns on DDS performance. We then explain the multi-stage

data processing pipeline that was developed to clean, transform, and structure the raw outputs from these campaigns. Finally, we conclude by presenting the curated datasets that are the primary output of this experimental work. This process uncovers non-trivial performance characteristics and provides the robust, empirical foundation required for the analytical work and predictive modelling that follows.

3.1.1 Contributions

The primary contributions of this chapter are as follows:

- The design and implementation of three distinct, controlled test environments (two physical and one virtualised) to facilitate reproducible DDS performance analysis.
- The development of a bespoke software suite for automated experiment execution, data collection, and preliminary analysis, enabling the systematic exploration of the vast DDS configuration space.
- The execution of over 20 systematic experimental campaigns, resulting in many large-scale, empirical datasets capturing DDS performance across thousands of configurations.
- The curation of the raw experimental data into 17 well-structured, analysis-ready datasets, which serve as the foundation for the subsequent analysis and modelling chapters.

3.2 Testbeds

To comprehensively evaluate the performance of DDS across different operational conditions, three distinct testbeds were established, as summarised in Table 3.1. This multi-environment approach ensures that our findings are robust and not confined to a single type of infrastructure. We used one virtualised environment to represent a common enterprise or cloud deployment, and two physical environments of different scales to provide a controlled, reproducible baseline for performance measurement.

3.2.1 Justification for Choice of Testbeds

The selection of testbeds for this research was guided by the need to balance realism with experimental control. The virtualised testbed was chosen to represent a typical, non-isolated enterprise or cloud environment, where DDS applications often coexist with other services and compete for resources. This setup, with its inherent network noise, provides a realistic setting for evaluating DDS performance under less-than-ideal conditions.

The physical testbeds, constructed from Raspberry Pi (RPi) clusters, were chosen for several reasons. Firstly, they provide a low-cost, power-efficient, and physically isolated environment, which is crucial for conducting controlled and reproducible experiments. The absence of unpredictable network noise in these clusters allows for a clearer measurement of the impact of specific DDS parameters on performance. Secondly, RPis, with their resource-constrained nature, are representative of many real-world embedded systems where DDS is deployed. By testing on this hardware, the findings of this thesis are more directly applicable to a wide range of practical use cases. While more powerful, industrial-grade hardware would offer a different performance profile, the use of RPis allows for a more accessible and replicable experimental setup, which is a key consideration for the broader research community.

Specification	Virtualised	3-Node Cluster (3Pi)	5-Node Cluster (5Pi)
Hardware	Shared Server	RPi 3A+	RPi 4B
CPU	Virtualised	1.4GHz	1.5GHz
RAM	8 GB	512 MB	4 GB
Operating System	CentOS 7.9	RPi OS	RPi OS
Network Interface	Virtualised	Ethernet	Ethernet

Table 3.1: Summary of Testbed Specifications

3.2.2 Virtualised Testbed

The first testbed comprised four Virtual Machine (VM) hosted on City, St George’s, University of London’s shared infrastructure. This setup was designed to emulate a typical, non-isolated enterprise environment where DDS applications might coexist with other services, competing for network and computational resources. Each VM was configured with CentOS 7.9-2009, allocated 8GB of RAM and a 250GB HDD. The machines operated on a Virtual Local Area Network (vLAN), meaning the network was shared with numerous other VMs and services. This inherent “network noise” was a key characteristic of this testbed, allowing for the evaluation of DDS performance under real-world conditions.

3.2.3 Physical Testbeds: Raspberry Pi Clusters

To complement the virtualised testbed, two physical testbeds were constructed using clusters of RPi devices. These clusters provided a low-cost, power-efficient, and physically isolated

environment, enabling highly controlled and reproducible experiments, free from the network unpredictability of the shared vLAN. This absence of unpredictable network noise was crucial for clearly measuring the impact on performance of DDS under controlled conditions. Both clusters followed a controller-slave architecture, where a single controller node managed the experiments on the slave nodes.

Key features common to both physical testbeds include:

- **Connectivity:** All nodes in each cluster were connected via a dedicated Ethernet switch to a private, isolated local network.
- **Control:** The controller node in each cluster (named **k1**) was equipped with cellular internet access via a router, allowing for remote management and data retrieval.
- **Reliability:** Each slave node was connected to a Tapo P100 smart plug, creating a remote power management system. This allowed for manual power cycling to recover any node that became unresponsive during long-running experiments, ensuring high availability.

The 3-Node Cluster (3Pi)

The smaller physical testbed consisted of one RPi 3B and two RPi 3A+ nodes. It included one controller (**k1**) and two slaves (**p1**, **p2**). This testbed served as a pathfinder, used for initial experiments and to validate the experimental tooling in a simple, controlled setting before deployment on the larger cluster.

Network Performance Characterisation To establish a baseline for the network’s maximum capacity, the **iperf** tool was used. The tests, run between all pairs of slave nodes, revealed a stable network with an average throughput of 5.5 Mbps and a maximum observed throughput of 9 Mbps, consistent with the capabilities of the 100 Mbps Ethernet ports on the RPi 3A+ under load. One might ask why a low value of 9 Mbps was observed instead of something closer to the theoretical maximum of 100 Mbps. This is due to the nature of the switch used in the testbed, which has 5 10/100/1000 Mbps ports and therefore dynamically negotiates the speed of each port based on the connected device.

The 5-Node Cluster (5Pi)

The larger physical testbed comprised five RPi 4B nodes (4GB model), with one controller (**k1**) and four slaves (**k2** through **k5**) as shown in Figure 3.1. This more powerful environment, with its Gigabit Ethernet capabilities, was designed to investigate the scalability of DDS performance

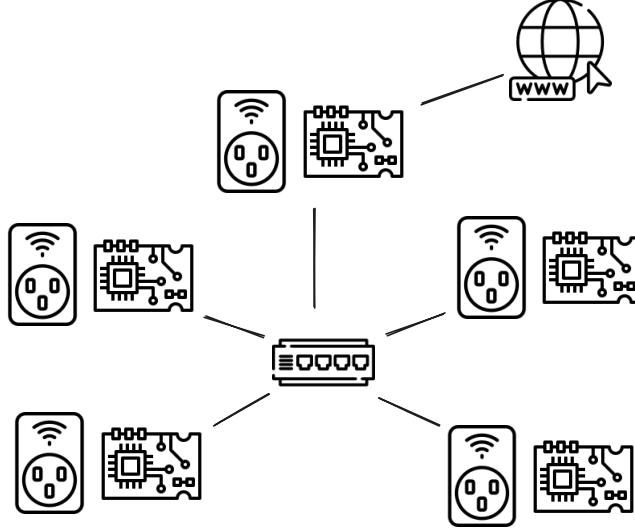


Figure 3.1: Physical 5-Node RPi Cluster Testbed

with an increased number of participating nodes.

Network Performance Characterisation Throughput tests using `iperf` were conducted across 12 permutations of client-server pairs among the slave nodes. Anomalous results were filtered and the results showed a robust average throughput of 94 Mbps. A single measurement of 1120 Mbps was recorded; however, this was dismissed as a measurement anomaly, as it exceeds the theoretical maximum of the Gigabit Ethernet interface and was not reproducible. The stable 94 Mbps figure was therefore taken as the representative network capacity.

3.3 Tools

To facilitate the extensive and complex experimentation required for this research, a suite of three primary software tools was employed. This suite consisted of a well-established benchmarking tool (Perftest), a custom-built automation framework to manage the experiments (AutoPerf), and a tool for post-experiment parameter analysis (PAT).

3.3.1 RTI Perftest

The core performance measurement was conducted using RTI Perftest¹, a command-line utility designed to benchmark the performance of DDS systems. It is a specialised tool for measuring

¹The source code and documentation for RTI Perftest are available at: <https://github.com/rticommunity/rtiperftest>

key metrics such as minimum latency, maximum throughput, and loaded latency across a wide range of configurable scenarios. This benchmarking tool was chosen after careful consideration of its capabilities, ease of use, and compatibility with the RTI Connex DDS implementation used in this research. Other benchmarking tools were found to be obsolete, unsupported, or lacking the necessary features for our requirements. This choice initiated the collaboration with RTI, which provided support and resources to enhance the tool’s functionality for our specific needs.

Perftest operates by creating two distinct application types: a Perftest Publisher and a Perftest Subscriber. To avoid ambiguity with the underlying DDS entities, we refer to these as *PPub* and *PSub*, respectively, while the standard DDS entities are referred to as *DPub* (DDS Publisher) and *DSub* (DDS Subscriber).

A crucial aspect of the Perftest architecture is that each of its applications contains both a DDS publisher and a DDS subscriber. That is, a single *PPub* application hosts one *DPub* and one *DSub*, and likewise for the *PSub* application. This internal structure allows the tool to manage the two-way communication necessary for metrics like round-trip time and latency measurement.

The tool allows the usage of two types of tests: latency or throughput. The latency tests are widely known as ping-pong tests where the publisher will send a latency measurement packet to the subscriber after taking a timestamp. Once the subscriber receives this latency measurement packet it returns it immediately and the publisher records the second timestamp after receiving the packet. This process is repeated continuously until either the defined total number of samples to be sent has been met or the test duration has been reached.

The throughput test is similar to the latency test. The publisher starts by taking a timestamp before sending a latency measurement packet (the same case as the latency test). Once the subscriber returns this latency measurement packet the round-trip time is recorded and the one-way latency is calculated by dividing this round-trip time by two. Once this measurement has been taken, the throughput test differs because the publisher then publishes a user-defined amount of throughput packets of a user-defined size. Once this amount of throughput packets has been communicated, the publisher will then send a latency measurement packet. Therefore, in summary, the latency test only sends latency measurement packets while the throughput test sends a latency measurement packet followed by a defined amount of throughput packets and the process is repeated. Figure 3.2 gives a visual comparison of the two types of performance tests. It should be noted that the packets shown in the figure are constant streams of communication.

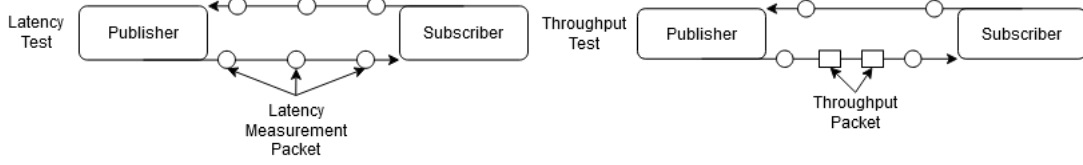


Figure 3.2: Latency and Throughput Test Comparison in RTI's Perftest.

The tool records the latency only on the first publisher. That is the publisher with an ID value of 0. This means that in the case where multiple publishers are present within the test, only the first publisher is recording the latency. With this in mind, it should be noted that this latency value is not exactly the “true” latency. The “true” latency would in this case be how long it takes the publisher to send a message to all subscribers. This includes the acknowledgement from all subscribers. However, since the tool describes that only the first publisher will send a single latency measurement packet to a single subscriber, the “true” latency measuring the transmission of a message is not recorded. Nonetheless, an aspect of the latency is still being recorded and we can use this metric to evaluate the performance of DDS. Javier Morales, the head of the team behind RTI Perftest assisted in the usage of the benchmarking tool throughout this work resulting in an ongoing collaboration with the developers of the leading DDS product for the past 5 years. He has assisted in understanding the granular functionality of the benchmarking tool and its configuration abilities. This collaboration has benefited both parties with discussion resulting in updates to the tool allowing functionality to write results to a file as well avoiding errors in undesired default configuration values which has increased the velocity of the experimental performance evaluation data collection stage.

3.3.2 AutoPerf

While RTI Perftest is effective for single tests, the scale of this research, requiring thousands of unique configurations, necessitated a robust automation solution. To this end, a bespoke automation tool named AutoPerf² was developed in Python. AutoPerf is a framework built around Perftest, designed to orchestrate the entire experimental lifecycle from test generation to data collection. The tool was developed using a modern software stack, including the use of `pytest` for a comprehensive test suite to ensure reliability.

AutoPerf's design is modular, allowing for flexibility and extensibility. Its key features include:

²The source code for AutoPerf is available at https://github.com/kaleem-peeroo/AutoPerf/tree/oop_refactor

- **Automated Testing:** Runs a large number of performance tests with minimal human intervention.
- **Flexible Configuration:** Defines complex test scenarios using a simple and intuitive Tom’s Obvious, Minimal Language (TOML) configuration file.
- **Resilient Operation:** Automatically retries failed tests, resumes interrupted campaigns, and handles unresponsive machines.
- **Remote Execution:** Runs tests on remote machines using Secure Shell (SSH), facilitating distributed experimentation.
- **Detailed Logging:** Tracks the status of each test in a comprehensive Experiment Status Spreadsheet (ESS).

Its features can be grouped into three key areas:

Test Definition and Generation: AutoPerf provides two methods for generating test configurations. To simulate imperfect network conditions, it was integrated with the standard Linux `tc` (traffic control) utility. Before each test run, AutoPerf constructed and executed the appropriate `tc` commands on the slave nodes, allowing for the dynamic application of specific network conditions.

Execution, Monitoring and Resilience: The tool manages the execution of test campaigns, which can be queued to run in sequence. State management and monitoring are handled through a mechanism named the ESS. For each campaign, an ESS file is created as a gzip-compressed JSONL file (`.jsonl.gz`), where each line represents a single test attempt with its configuration, timestamps, and outcome, including any errors. This structured log serves two critical functions: firstly, it provides a detailed audit trail for monitoring and debugging. Secondly, it enables robust resilience; if a campaign is interrupted, AutoPerf consults the ESS to identify the last successful test and resumes the campaign from that point, preventing data loss and redundant work. The tool also provides real-time feedback to the user’s console using the `rich` library and can automatically power-cycle unresponsive nodes using the `tapo` library.

Data Management: After each test run, AutoPerf automatically retrieves the output, parses the key performance metrics using the `pandas` and `numpy` libraries, and aggregates the results. The final, structured datasets are stored in the efficient Apache Parquet (*Parquet* n.d.) format using the `fastparquet` library.

Configuration

Every experimental campaign is defined by a central configuration file using the TOML format, which is parsed using the `toml` library. Configuration files are structured with one or more `[[campaigns]]` sections, each defining a set of experiments. This file dictates the behaviour of AutoPerf, specifying the test generation strategy, the QoS parameters to be explored, the network noise settings, and the machine details for the testbed nodes. An example of such a configuration file is shown in Listing 3.1.

```
[[ campaigns ]]
campaign_name = "My First Campaign"
gen_type = "pcg"
max_failures = 10
max_retries = 3
restart_after_retries = 2
ping_attempts = 3
ssh_attempts = 3

[campaigns.qos_settings]
duration_secs = [10, 20]
datalen_bytes = [1024, 2048]
pub_count = [1, 2]
sub_count = [1, 2]
use_reliable = [true, false]
use_multicast = [true, false]
durability = [0, 1]
latency_count = [100]

[[ campaigns.slave_machines ]]
hostname = "machine1"
participant_type = "pub"
ip = "192.168.1.101"
ssh_key_path = "~/.ssh/id_rsa"
username = "user"
perfctest_path = "/path/to/perfctest_publisher"

[[ campaigns.slave_machines ]]
```

```

hostname = "machine2"
participant_type = "sub"
ip = "192.168.1.102"
ssh_key_path = "~/.ssh/id_rsa"
username = "user"
perftest_path = "/path/to/perftest_subscriber"

```

Listing 3.1: Example AutoPerf configuration file in toml format.

Architecture and Workflow

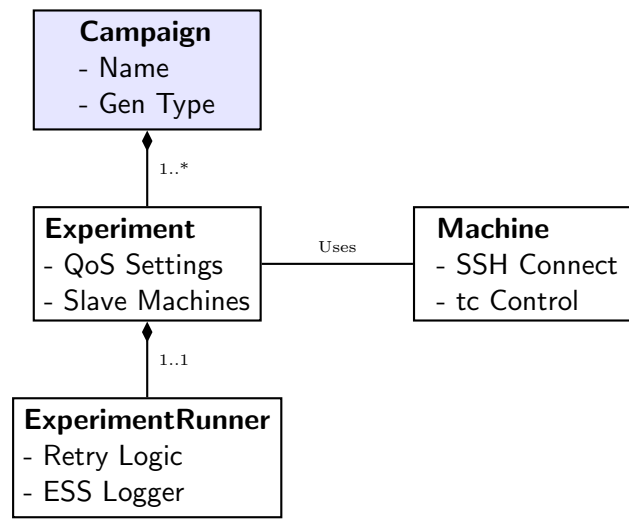


Figure 3.3: Modular class architecture of the AutoPerf framework.

AutoPerf is built with a modular architecture. Figure 3.3 shows several key classes that manage different aspects of the experimental process:

- **Config:** Responsible for parsing and validating the TOML configuration file, creating **Campaign** objects.
- **Campaign:** Represents a collection of experiments, managing experiment generation (Pre-set Combination Generation (PCG) or Random Combination Generation (RCG)) and the ESS.
- **Experiment:** Encapsulates a single test with a specific QoS configuration and a set of machines.
- **ExperimentRunner:** Orchestrates the execution of a single experiment, handling environment setup, test execution, and result collection.
- **Machine:** Represents an individual testbed machine (publisher or subscriber), managing its SSH connection (using **paramiko**), script execution, and result retrieval.

- **QoS**: Represents the QoS settings for an experiment.
- **SSHClient**: A dedicated wrapper around the `paramiko` library for robust SSH connections.

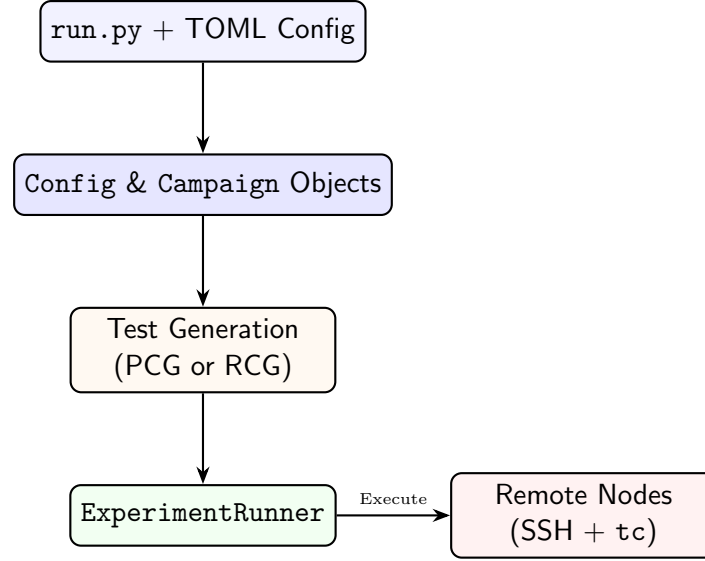


Figure 3.4: The AutoPerf experimental lifecycle and data flow.

The typical workflow for an AutoPerf campaign shown in Figure 3.4 is as follows:

1. The main execution script (e.g., `run.py`) is invoked with a path to the configuration file.
2. The `Config` class parses the configuration and instantiates one or more `Campaign` objects.
3. Each `Campaign` generates a list of `Experiment` objects based on the specified generation type (PCG or RCG).
4. For each `Experiment`, an `ExperimentRunner` object is created and executed.
5. The `ExperimentRunner` connects to the designated `Machine` instances, runs the Perftest benchmarks, and collects the raw results.
6. The results, along with experiment metadata and any errors, are recorded in the ESS.

Preset Combination Generation

The PCG mode is designed for exhaustive, systematic exploration of a defined parameter space. When `gen_type` is set to `pcg`, AutoPerf computes the Cartesian product of all parameter lists provided in the `qos_settings`. This generates a test for every possible combination of the specified parameters, ensuring complete coverage of the defined experimental space. This mode is ideal for smaller-scale experiments where comprehensive testing is feasible.

Random Combination Generation

The RCG mode is designed for the efficient exploration of a large parameter space where exhaustive testing would be computationally intractable. In this mode, AutoPerf uniformly randomly samples one value from each parameter list in `qos_settings` to construct a unique test configuration. This process is repeated until a `rcg_target_test_count` of unique configurations has been generated. This stochastic approach allows for broad coverage of a vast configuration space, increasing the likelihood of discovering non-obvious parameter interactions.

3.3.3 AutoPerf Monitor

To provide real-time oversight and facilitate post-campaign analysis of the extensive experimental data, a dedicated companion tool, the AutoPerf Monitor, was developed. Its primary purpose is to monitor ongoing AutoPerf campaigns running on remote clusters, automating the collection of performance data and logs.

Architecture and Functionality

AutoPerf Monitor operates by connecting to remote clusters via SSH, using the `paramiko` library for secure communication. It is configured via TOML files (e.g., `3pi.toml`, `5pi.toml`) that define the remote clusters, including their IP addresses, SSH credentials, and the path to the AutoPerf installation on the remote machine.

The core functionality of the Monitor includes:

- **Remote Data Collection:** Automatically downloads the latest ESS files and restart logs (e.g., `tapo_restart.log`) from the remote AutoPerf instances.
- **Status Reporting:** Provides a Command Line Interface (CLI) for reading and displaying ESS data in a human-readable format, leveraging the `rich` library for enhanced console output and tables. It can parse and highlight errors and standard output from machine runs.
- **Campaign Management:** Allows users to select a specific cluster and ESS file for detailed inspection, aiding in debugging and progress tracking.

Key components of the AutoPerf Monitor include:

- **Config:** Parses and validates the TOML configuration files for remote clusters.
- **Cluster:** Encapsulates operations related to a single remote cluster, managing SSH communication and file transfers.

- **SSHClient:** A wrapper around `paramiko` for simplified SSH operations (connecting, executing commands, file listing, downloading).
- **ESS:** Handles the parsing, manipulation (using `pandas.DataFrame`), and reporting of downloaded ESS files.

3.3.4 Performance Analysis Tool

The final component of the bespoke software suite was a Python-based CLI tool designed for in-depth analysis of the experimental data. Its primary goal is to assist in understanding how different experimental parameters influence performance metrics, and to identify optimal parameter settings.

Key Features and Functionality

The Performance Analysis Tool (PAT) offers a comprehensive set of features for data exploration and statistical analysis:

- **Data Loading and Preparation:** Loads data from a specified Parquet file (e.g., `2025_02_21_datasets.parquet`), standardises experiment names, and allows for the exclusion of undesired datasets.
- **Interactive Filtering:** Provides an interactive command-line interface for dynamically filtering datasets based on specific parameter values, enabling focused analysis of subsets of the data.
- **Statistical Rigour:** Employs statistical methods (e.g., Kolmogorov-Smirnov (KS)) test and confidence intervals for robust comparisons of data distributions.
- **Advanced Visualisation:** Generates Empirical Cumulative Distribution Function (eCDF) plots with confidence bands to visually represent performance differences between parameter combinations. It supports both static plots (using `matplotlib.pyplot` and `seaborn`) and terminal-based plotting (using `plotext`).
- **Automated Classification:** Identifies and classifies the most performant parameter combinations based on statistical analysis.
- **Correlation Analysis:** Utilises Chi-squared (χ^2) tests and Cramér's V (V) to determine correlations between parameters and performance outcomes.

Operational Mechanism

A summary of how the PAT operates is as follows:

1. Generate variations of parameter value, e.g., 100 B + 1 pub vs. 100 B + 10 pubs vs. 100 B + 15 pubs.
2. Generate all pairings of variations, e.g., 1 pub vs. 10 pubs, 1 pub vs. 15 pubs, 10 pubs vs. 15 pubs.
3. For each pairing:
 - (a) Let the pairing be made up of **a** and **b**.
 - (b) Calculate eCDF of **a** and eCDF of **b**.
 - (c) Calculate error margin of **a** and error margin of **b**.
 - (d) Find all matching probabilities in both eCDF of **a** and eCDF of **b** such that each probability is accurate to 2 decimal places, where the probabilities are point estimates corresponding to quantile values as calculated in the library (*ecdf* — *SciPy v1.16.2 Manual* n.d.).
 - i. Example:
 - ii. If `probabilities_a` = [0.0003, 0.0005, 0.0006, 0.01412, 0.02344, 0.1541].
 - iii. If `probabilities_b` = [0.0001, 0.0003, 0.0005, 0.05169, 0.15412].
 - iv. Then the matching probabilities would be [0.00, 0.05, 0.15].
 - (e) Remove the quantile values for the matching probabilities.
 - (f) You should be left with a single list of probabilities to 2 decimal places and quantile values for **a** and quantile values for **b** for each probability value seen in the matching probabilities.
 - (g) Interpolate for all missing intermediary values in either list.
 - (h) For each quantile in both **a** and **b**, calculate the lower and upper values by subtracting and adding the error margin.
 - (i) Calculate the difference between the lower value of **b** and the upper value of **a** for all values.
 - (j) If all differences are positive, then **a** is on the left and **b** is on the right.
 - (k) Calculate the difference between the lower value of **a** and the upper value of **b**.
 - (l) If all differences are positive, then **b** is on the left and **a** is on the right.
 - (m) Note this classification down for the given pairing, i.e., record who is leftmost between **a** and **b**, who is rightmost between **a** and **b**, or if they cross over.
4. At this point, you are left with a list of pairings and the leftmost and rightmost value per pairing.
5. If interested in latency:

- (a) Find the value that appears as leftmost most of the time and assign as temporary winner.
- 6. If interested in throughput:
 - (a) Find the value that appears as rightmost most of the time and assign as temporary winner.
- 7. If the winner has ever intersected with any other value, there is no winner; else, this is the winner.

A sketch of the eCDF stochastic ordering algorithm is shown in Figure A.1 in Appendix A.

3.4 Experimental Campaigns

3.4.1 Introduction

This section details the systematic experimental campaigns conducted to comprehensively evaluate the performance of the DDS. These investigations progressively explore DDS behaviour across diverse configurations and environmental conditions, building from foundational studies to more complex, real-world scenarios. Our primary objective is to elucidate the intricate relationships between DDS performance metrics (such as latency and throughput) and key influencing factors, including participant numbers, communication modes, data message sizes, network noise, and QoS settings. By executing these campaigns across distinct testbeds, we aim to provide robust, generalisable insights into DDS scalability and reliability.

3.4.2 Justification for Choice of Experimental Parameters

The experimental campaigns were designed to systematically explore the vast configuration space of DDS. The choice of parameters and their ranges was guided by a combination of factors, including the findings of the literature review, the capabilities of the testbeds, and the desire to investigate both typical and extreme use cases.

The number of publishers and subscribers was varied to assess the scalability of DDS. The chosen ranges, from 1 to 75, cover a wide spectrum of deployment scenarios, from simple one-to-one communication to large-scale, many-to-many interactions. The data message sizes were selected to represent a variety of application payloads, from small control messages (e.g., 100 B) to larger data packets (e.g., 1 MB).

The QoS parameters, such as reliability and durability, were chosen for their known impact on performance. Both reliable and best-effort communication were tested to evaluate the

trade-off between guaranteed delivery and performance. The different levels of durability were investigated to understand the performance implications of data persistence. The use of MC was also a key parameter, as it is a common optimisation for one-to-many communication.

The introduction of network noise, through Bandwidth (BW) capping, was a deliberate choice to move beyond idealised laboratory conditions and to evaluate DDS performance in more realistic, constrained environments. The chosen BW caps were selected to represent different levels of network congestion, from moderate to severe.

While it is impossible to cover the entire DDS configuration space, the selected parameters and their ranges provide a representative and comprehensive basis for the performance evaluation and prediction tasks undertaken in this thesis.

Contributions

This section presents a series of pivotal experimental campaigns, each contributing uniquely to our understanding of DDS performance:

- Initial studies established a crucial baseline for DDS performance on VMs, exploring fundamental characteristics.
- Further investigations on VMs focused on MC communication, Distributed Denial of Service (DDoS) simulations, and malicious subscriber attacks.
- The impact of single versus multi-machine deployments was also rigorously examined.
- On the 3Pi Cluster, campaigns precisely determined participant thresholds and re-evaluated MC performance in a controlled physical environment.
- Significant effort was dedicated to simulating network noise through BW capping on the 3Pi Cluster, facilitating the collection of data for Machine Learning (ML) model training.
- The 5Pi Cluster extended these investigations, including comprehensive QoS capture campaigns with the sole purpose of exploring the QoS parameter space.
- Studies on increasing load and further participant threshold analyses were conducted on the 5Pi testbed.
- Additional noise generation experiments were performed on the 5Pi testbed to enrich the dataset for predictive modelling.

3.4.3 Virtualised Environment

Base Case

To establish a foundational performance baseline against which all subsequent, more complex scenarios could be compared, a “Base Case” campaign was designed. The primary objective was to quantify the fundamental throughput and latency characteristics of DDS in a controlled, minimally configured environment on the virtualised testbed.

The experimental setup involved varying the number of publishers and subscribers, and the communication mode (UC or MC). Key parameters were:

- **Publishers and Subscribers:** Either 1 or 3 publishers, and either 1 or 3 subscribers.
- **VM Allocation:** For each VM, publishers were allocated as 1 on VM1, 1 on VM2, 1 on VM3, 0 on VM4, and subscribers as 1 on VM1, 1 on VM2, 1 on VM3, 0 on VM4.
- **Data Message Size:** Fixed at 100 B.
- **Test Duration:** Each test ran for 21600 seconds (6 hours).
- **Reliability:** All communication was configured to be reliable, ensuring guaranteed message delivery.
- **Communication Mode:** Both UC and MC communication were tested.

This experiment serves as a crucial baseline against which all subsequent experimental campaigns on the VMs will be compared.

Multicast Exploration

Building upon the established baseline, this campaign was designed to test a core hypothesis regarding DDS scalability. It posits that MC communication offers a significant performance advantage over UC as the number of subscribers increases. While UC requires the publisher to send a separate copy of each message to every subscriber, MC allows a single message to be sent to a group address. This theoretically reduces network traffic and publisher load. This experiment aimed to quantify this effect by systematically measuring latency and throughput across a wide range of publisher and subscriber counts for both communication modes. We set up our tests with the values seen in Table 3.2. Results of this campaign were published in (Peeroo et al. 2022).

Setting	Values
Data Message Size	100 B
Test Duration	6 Hours
Latency Count	1000
Reliability	Reliable
Publisher Count	1, 3, 10, 25, 50, 75
Subscriber Count	1, 3, 10, 25, 50, 75

Table 3.2: MC Exploration Campaign Configuration. Latency count is the number of messages to send before a latency measurement is taken. In this case latency measurements are taken every 1000 messages.

Distributed Denial of Service

To investigate the resilience of DDS to network saturation, a campaign was designed to simulate a DDoS attack. The hypothesis was that high-volume traffic, even on a separate DDS domain, could degrade the performance of a primary application by consuming shared network resources. This scenario models a common security concern in shared network environments. The campaign compared a baseline of normal DDS traffic against a condition where a high-load “attacker” domain was simultaneously active, allowing for a direct measurement of the performance degradation caused by network contention.

We varied the following aspects:

- **Publisher and Subscriber Counts:** 20, 50, 100, and 150 pairs of publishers and subscribers.
- **Normal Usage Simulation:** Half of these simulated normal network usage, with a data message size of 100 B.
- **DDoS Attack Simulation:** The other half simulated a DDoS attack, where we varied the data message size as follows:
 - 300 B, 500 B, 1 KB, 16 KB, 64 KB, 128 KB, 512 KB, and 1 MB.

This experiment allows for direct comparison with the Base Case Experimental Campaign because both experiments used the same number of non-malicious publishers and subscribers.

Malicious Subscriber Attack

Setting	Values
Data Message Size	100 B
Test Duration	900 Seconds
Latency Count	1000
Reliability	Reliable

Table 3.3: Malicious Subscriber Attack Campaign Configuration

This campaign was designed to investigate a different type of security vulnerability: the resource-depletion attack. The hypothesis was that a set of “malicious” subscribers, by expressing interest in data without ever reading it, could consume publisher resources and degrade performance for legitimate subscribers. This scenario models a subtle attack vector where the network itself is not saturated, but the publisher’s internal queues and memory are strained. The experiment measured the performance of a group of “good” subscribers while a varying number of “bad” subscribers were simultaneously connected to the same topic. Configurations for this campaign are shown in Table 3.3 for the following publisher/subscriber combinations:

- 10 publishers and 20 subscribers
- 25 publishers and 50 subscribers
- 50 publishers and 100 subscribers
- 75 publishers and 150 subscribers

Confirmation of Variation

A foundational methodological question for this entire body of research was the determination of a sufficient experiment duration. An adequate duration must be long enough to overcome initial transient effects and produce statistically stable, reproducible results, without being so long as to make the execution of thousands of tests computationally infeasible. To this end, a dedicated campaign was conducted to quantify the impact of test duration on performance measurement variability. Initial exploratory tests ran for 15 minutes yielded inconsistent results with high variance. The test duration was therefore extended, with a 12-hour test demonstrating significantly more stable measurements. Based on these findings, a duration of 6 hours was selected as the standard for all subsequent campaigns on the VMs, representing a robust trade-off between the need for statistical stability and the practical constraints of large-scale experimentation. This campaign involved four experiments. We varied the following aspects:

- **Publisher/Subscriber Counts:** 1 and 3 publishers, and 1 and 3 subscribers.
- **VM Allocation:** Publishers were allocated per VM as 1, 1, 1, 0, and subscribers as 1, 1, 1, 0.
- **Data Message Size:** Fixed at 100 B.
- **Test Duration:** Each test ran for 21600 seconds (6 hours).
- **Reliability:** All communication was configured to be reliable, ensuring guaranteed message delivery.
- **Communication Mode:** Both UC and MC were tested.

Single vs Multi Machine

This campaign was designed to investigate the performance implications of co-locating DDS participants on the same physical host. The central hypothesis is that communication between participants on the same machine may occur via a more efficient intra-process or shared-memory transport, bypassing the network stack entirely and resulting in significantly lower latency compared to communication between participants on separate machines. This experiment aimed to test this hypothesis by measuring the latency of two scenarios: one where the publisher and subscriber are on the same host, and one where they are on different hosts, thereby clarifying whether DDS is intelligent enough to optimise for co-located participants.

3.4.4 3 Raspberry Pi Cluster

QoS Capture

Following the initial investigations on the virtualised testbed, the research moved to the controlled environment of the 3Pi cluster. The primary objective of this phase was to conduct a large-scale, systematic exploration of the DDS QoS parameter space. This “QoS Capture” campaign was designed to generate a comprehensive dataset for a fine-grained analysis of how different QoS policies and their interactions influence performance. The central hypothesis is that QoS settings related to Reliability and Durability will have a significant and quantifiable impact on both latency and throughput, and that these impacts may not always be linear or intuitive. This dataset serves as a foundational resource for the detailed performance evaluation in Chapter 5. The tests were configured with the settings seen in Table 3.4.

Setting	Values
Duration (s)	600
Data Length (B)	100, 512, 1024, 4096, 8192
Publisher Count	1, 5, 10, 15, 20, 25
Subscriber Count	1, 5, 10, 15, 20, 25
Reliability	true, false
MC	true, false
Durability Level	0, 1, 2, 3
Latency Count	100

Table 3.4: 3Pi QoS Capture Campaign Configuration

Participant Threshold

Before conducting large-scale experiments on the 3Pi cluster, it was essential to first establish the practical limits of the testbed itself. This “Participant Threshold” campaign was therefore designed as a prerequisite investigation to determine the maximum number of DDS participants (publishers and subscribers) that the RPi 3A+ nodes could reliably sustain. The hypothesis was that beyond a certain number of participants, resource limitations (such as CPU, memory, or network interface capacity) would lead to failures, dropped samples, and unresponsive applications, rendering the performance data invalid. By systematically increasing the participant count, this campaign aimed to identify this operational ceiling, thereby ensuring that all subsequent experiments on this testbed were designed to run within its known reliable limits. The tests were configured with the settings seen in Table 3.5.

Setting	Values
Duration (s)	600
Data Length (B)	512
Publisher Count	TBD
Subscriber Count	TBD
Reliability	true, false
MC	true, false
Durability Level	3
Latency Count	100
BW Cap	-

Table 3.5: Participant Threshold Campaign Configuration

Through iterative testing, it was determined that 26 publishers and 26 subscribers represented the operational threshold for this testbed.

Multicast Exploration

Having established the operational limits of the 3Pi cluster, this campaign aimed to re-evaluate the performance of MC communication in a controlled, physical environment. The central hypothesis, based on previous findings, was that UC communication would outperform MC at higher participant counts, contrary to the theoretical efficiency of the MC protocol. It is hypothesised that this effect is due to implementation-specific overheads or resource contention that emerges at scale. This campaign was therefore designed to systematically compare the two communication modes up to the maximum reliable participant count of the testbed, which was established as 26 participants in the preceding “Participant Threshold” campaign. The tests were configured with the settings seen in Table 3.6.

Setting	Values
Duration (s)	600
Data Length (B)	100, 512
Publisher Count	1, 5, 10, 15, 20, 25
Subscriber Count	1, 5, 10, 15, 20, 25
Reliability	true
MC	true, false
Durability Level	0
Latency Count	100

Table 3.6: 3Pi Multicast Exploration Campaign Configuration

Participant allocation involved one host for publishers and one host for subscribers.

BW Capping

To enhance the ecological validity of our experimental setup, this campaign was designed to investigate the impact of simulated network noise on DDS performance through the addition of limitations on the BW. Recognising that isolated network environments are not generally representative of real-world deployments, the central hypothesis was that controlled network impairments, specifically BW limitations and packet-level corruptions, would significantly affect DDS latency and throughput. This campaign employed the Linux *netem* utility, integrated within *AutoPerf*, to introduce caps on the available BW and to simulate various network noise profiles. The objective was to generate a dataset reflecting DDS behaviour under realistic, non-ideal network conditions, thereby providing a more robust foundation for performance prediction.

To train ML models for predicting DDS performance under more realistic usage, a substantial amount of data is required. This data needs to encompass various levels of simulated network interference.

Specifically, we aimed to collect approximately 1000 tests with minimal noise, corresponding to an 8 Mbps BW cap. Additionally, 1000 tests with significant noise were targeted, corresponding to a 6 Mbps BW cap.

Based on previous findings on the 3Pi testbed, the lowest achievable BW throttle was 5 Mbps. Conversely, the maximum throughput observed on the 3Pi testbed was 9 Mbps.

Assuming a minimum throughput of 5 Mbps and a maximum of 9 Mbps, the 25% value is approximately 6 Mbps, and the 75% value is 8 Mbps. These values informed the selection of two distinct campaigns for noise levels:

- The Low BW campaign.

- The High BW campaign.

Low BW (6 Mbps)

This campaign, designated “ML Data Collection 6Mbps”, was a targeted effort to gather a substantial dataset specifically for training ML models. The primary objective was to enable the prediction of DDS performance under a defined low BW noise profile, precisely set at 6 Mbps. This dataset is crucial for understanding DDS behaviour when network capacity is significantly constrained, a common scenario in many real-world deployments. The tests were configured with the settings seen in Table 3.7. The duration of 600 seconds is significantly shorter than the 6 hours used in the VM campaigns. This choice of a shorter duration was a result of multiple factors. First, the results from the VM campaigns indicated that 6 hours was necessary to achieve stable and reproducible results in a virtualised environment. This is different to the 3Pi and 5Pi testbeds, where there are no processes sharing the network. The 3Pi and 5Pi testbeds are designed to run isolated experiments, where each test is conducted in a controlled environment without the overhead of virtualisation. Therefore, with less variation seen in shorter durations, we opted for a 600-second duration for the 3Pi and 5Pi testbeds.

Setting	Values
Duration (s)	600
Data Length (B)	512, 1000
Publisher Count	1, 5, 10, 15, 20
Subscriber Count	1, 5, 10, 15, 20
Reliability	true, false
MC	true, false
Durability Level	0, 1, 2, 3
Latency Count	100
BW Cap	6 Mbps

Table 3.7: Low BW Cap Campaign Configuration (6 Mbps)

High BW (8 Mbps)

This campaign, designated “ML Data Collection 8Mbps”, was a parallel effort to the low BW cap campaign, also aimed at gathering a substantial dataset for training ML models. The objective was to predict DDS performance under a specific high BW noise profile, precisely set at 8 Mbps. This dataset complements the low BW data by providing insights into DDS behaviour under less severe, but still constrained, network conditions, thereby enriching the training data for robust predictive models. The tests were configured with the settings seen in Table 3.8.

Setting	Values
Duration (s)	600
Data Length (B)	512, 1000
Publisher Count	1, 5, 10, 15, 20
Subscriber Count	1, 5, 10, 15, 20
Reliability	true, false
MC	true, false
Durability Level	0, 1, 2, 3
Latency Count	100
BW Cap	8 Mbps

Table 3.8: High BW Cap Campaign Configuration (8 Mbps)

3.4.5 5 Raspberry Pi Cluster

QoS Capture

This campaign, designated “QoS Capture”, was meticulously designed to amass a comprehensive dataset. This dataset encompasses a wide spectrum of QoS configurations on the 5Pi testbed. The primary objective was to generate sufficient empirical data. This data is crucial for training robust ML models. These models are intended to predict DDS performance across diverse QoS settings. The 5Pi testbed, with its increased node count compared to the 3Pi cluster, allowed for the investigation of DDS behaviour under higher participant loads and more complex communication patterns. This systematic exploration is vital for understanding the scalability and performance characteristics of DDS in larger, distributed environments. The campaign was executed in three distinct variations. Each variation differed in the strategic allocation of publishers and subscribers across the 5Pi testbed machines. These variations were: 1P 3S, 2P 2S, and 3P 1S. Here, “P” denotes machines exclusively hosting publishers, and “S” denotes machines exclusively hosting subscribers.

Setting	Values
Duration (s)	600
Data Length (B)	100
Publisher Count	1, 5, 10, 25
Subscriber Count	1, 5, 10, 25
Reliability	true, false
MC	true, false
Durability	0, 1, 2, 3

Table 3.9: QoS Capture Configuration for 1P 3S, 2P 2S, and 3P 1S

Increasing Load

This campaign investigated the behaviour of DDS when increasing the load by varying the allocation of publishers and subscribers across the 5Pi Cluster’s four slave machines. Specifically, it explored scenarios where all publishers were placed on a single machine, while subscribers were distributed across the remaining machines.

Three primary participant allocation scenarios were examined:

- **1P1S:** One machine hosted all publishers, and one machine hosted all subscribers.
- **1P2S:** One machine hosted all publishers, and two machines hosted all subscribers.
- **1P3S:** One machine hosted all publishers, and three machines hosted all subscribers.

Where “XPYS” denotes that X machines hosted all publishers and Y machines hosted all subscribers. The tests were configured with the settings seen in Table 3.10.

Setting	Values
Duration (s)	600
Data Length (B)	100, 1000
Publisher Count	1, 5, 10, 25
Subscriber Count	1, 5, 10, 25
Reliability	true, false
MC	true, false
Durability Level	0
Latency Count	100

Table 3.10: Increasing Load Campaign Configuration for 1P1S, 1P2S, and 1P3S

Participant Threshold

This campaign aimed to determine the maximum number of publishers and subscribers that could be reliably run on the 5Pi Cluster testbed. It was hypothesised that the participant limit would be reached at some point after 20 publishers or subscribers. Therefore, testing commenced from 15 participants. The tests were configured with the settings seen in Table 3.11.

Through iterative testing, it was determined that 20 publishers and 20 subscribers represented the operational threshold for reliable usage on this testbed. Best effort communication, however, was observed to handle up to 28 publishers and subscribers.

Multicast Exploration

This campaign aimed to confirm the results previously observed regarding the effects of MC communication on DDS performance, as detailed in a prior publication (Peeroo et al. 2022).

Setting	Values
Duration (s)	600
Data Length (B)	512
Publisher Count	TBD
Subscriber Count	TBD
Reliability	true, false
MC	true, false
Durability Level	3
Latency Count	100
BW Cap	-

Table 3.11: 5Pi Participant Threshold Campaign Configuration

Specifically, it sought to verify the finding that UC communication somewhat uninventively as we reported in (Peeroo et al. 2022) outperforms MC for a larger number of participants. These tests were replicated on the 5Pi Cluster testbed. Prior to this campaign, the Participant Threshold campaign was conducted to determine the maximum number of publishers and subscribers the 5Pi Cluster could handle. This preliminary work concluded that 20 publishers and 20 subscribers represented the maximum reliable participant count for this testbed. It was hypothesised that UC would outperform MC for larger numbers of participants. The tests were configured with the settings seen in Table 3.12.

Setting	Values
Duration (s)	600
Data Length (B)	512
Publisher Count	1, 5, 10, 15, 20
Subscriber Count	1, 5, 10, 15, 20
Reliability	true
MC	true, false
Durability Level	0
Latency Count	100

Table 3.12: 5Pi Multicast Exploration Campaign Configuration

Two participant allocation strategies were employed:

- 1 machine for publishers and 3 machine for subscribers (1P3S).
- 2 machines for publishers and 2 machines for subscribers (2P2S).

Noise Generation

This campaign aimed to confirm the effectiveness of network noise generation by throttling BW on the 5Pi Cluster. It sought to verify that varying levels of simulated noise would indeed impact DDS performance. To define appropriate BW throttling values, the average (94 Mbits/sec) throughput of the 5Pi Cluster was considered. Based on these figures, tests were designed to

run at 24 Mbps (25% of max) and 71 Mbps (75% of max). All tests used the QoS settings seen in Table 3.13, with only the BW cap varying. All tests were conducted using four slave machines with two machines dedicated to publishers and two machines dedicated to subscribers.

Setting	Values
Duration (s)	600
datalen (B)	512
Publisher Count	1, 5, 10, 15, 20
Subscriber Count	1, 5, 10, 15, 20
Reliability	true, false
MC	true, false
Durability Level	3
Latency Count	100
BW Cap	24, 47, 71

Table 3.13: 5Pi Noise Generation Campaign Configuration

ML

This campaign, titled “ML Data Collection”, aimed to gather a substantial dataset for training ML models. The objective was to predict DDS performance under varying levels of network noise. Specifically, the campaign sought to collect approximately 1000 tests with significant noise, corresponding to a 25% BW usage (24 Mbps). Additionally, 1000 tests with minimal noise were targeted, corresponding to a 75% BW usage (71 Mbps). The tests were configured with the settings seen in Table 3.14.

Setting	Values
Duration (s)	600
Data Length (B)	512, 1000
Publisher Count	1, 5, 10, 15, 20
Subscriber Count	1, 5, 10, 15, 20
Reliability	true, false
MC	true, false
Durability Level	0, 1, 2, 3
Latency Count	100
BW Cap	24 Mbps, 71 Mbps

Table 3.14: ML Data Collection Campaign Configuration

Participant allocation involved two hosts for publishers and two hosts for subscribers. The data collected for each BW cap was separated into two distinct datasets.

3.5 Data Processing

This section details the comprehensive methodology employed for processing the raw experimental data. It outlines the transformation of raw files into structured datasets, addresses potential data anomalies, and describes the techniques used for filtering faulty experiments. Furthermore, this section elaborates on the transient and outlier analysis procedures applied to ensure data quality and reliability. The section is structured as follows:

- Subsection 3.5.1 describes the conversion of raw experimental files into structured datasets, detailing the format and content of the files.
- Subsection 3.5.2 explains the criteria and methods used to identify and remove faulty experiments from the datasets.
- Subsection 3.5.3 outlines the two-stage process for transient and outlier analysis, ensuring the integrity of the data used for further analysis.

3.5.1 Raw Files to Datasets

The process of parsing the raw files into datasets begins with each experiments which generates a `csv` file for the initial publisher and a separate `csv` file for each subscriber. Consequently, an experiment involving n subscribers is expected to yield $n + 1$ `csv` files.

For latency measurements, the `csv` file originating from the primary publisher is designated as `pub_0.csv`. This file comprises the following columns:

- Length (B)
- Latency (μs)
- Ave (μs)
- Std (μs)
- Min (μs)
- Max (μs)

Meanwhile, each subscriber's `csv` file is designated as `sub_n.csv`, where n represents the subscriber's identifier. Each subscriber file contains the following columns:

- **Length (B)**: The size of the DDS message in B.
- **Total Samples**: The total number of samples received by the subscriber.
- **Samples/s**: The number of samples received per second.
- **Avg Samples/s**: The average number of samples received per second.

- **Mbps**: The throughput measured in megabits per second.
- **Avg Mbps**: The average throughput measured in megabits per second.
- **Lost Samples**: The number of samples that were not received by the subscriber.
- **Lost Samples (%)**: The percentage of samples that were not received by the subscriber.

Each `sub_n.csv` file is expected to contain x rows, where x corresponds to the experiment's duration in seconds.

File Processing

The processing of these raw files necessitates reading the file content to identify the designated start and end messages. Subsequently, these delimiting sections are removed, and all intermediate measurements are parsed. Only the latency measurements are extracted, with all associated calculated statistics being disregarded, as these can be re-derived post-processing. This procedure yields a singular list of latency measurements, which is then stored in a dedicated column within a Parquet file, alongside throughput measurements in other columns.

We extract all relevant columns from the subscriber files, including the Mbps column, which is subsequently stored in a separate Parquet file dedicated to that specific experiment which also contains latency measurements and throughput measurements for all other subscribers. The measurement processing involves iterating through each subscriber file. The start message and the initial occurrence of the end message are identified. All measurements situated between these two points are then processed. The subscriber's identifier is prepended to the Mbps column within the experiment's Parquet file e.g., `sub_n_mbps`. Upon loading all subscriber data into the experiment's Parquet file, the average and total Mbps across all subscribers are computed on a per-row basis. This entails iterating through each row, aggregating all samples per subscriber, and then calculating both their average and sum. These computations yield two distinct columns: average Mbps per subscriber and total Mbps across all subscribers. This is repeated for other metrics seen in the subscriber files, such as total samples, samples/s, avg samples/s, lost samples, and lost samples (%).

Average versus Total Throughput

Throughput is measured on a per-second basis for each subscriber. Experiments vary in the number of active subscribers. To facilitate comparisons across different experiments, the total throughput received across all subscribers is computed. While the calculation of average

throughput across all subscribers was considered, this metric does not readily permit straightforward inter-experiment comparisons.

Potential Data Anomalies

Certain experimental data may exhibit anomalies. In some instances, latency measurements are entirely absent. The corresponding files indicate both the initiation and termination of the experiment, yet no measurements are recorded. Observations suggest that such occurrences correlate with experimental conditions leading to exceptionally high latencies, thereby precluding the acquisition of a sufficient number of measurements within the designated experiment duration. It is important to note that latency measurements are captured at intervals of m messages, as specified by the latency count setting and explained in Subsection 3.3.1. Consequently, a combination of a high latency count and significantly large latency values is likely to result in no measurements being recorded during the experiment. Furthermore, some latency `csv` files are found to be completely empty, indicated by a data length of 0 B. These instances are classified as faulty experiments. The emptiness of these files may stem from the experiment becoming unresponsive. It is also possible that the experiment failed to conclude properly. The precise cause of this behaviour remains undetermined. In these specific cases, subscriber throughput files typically contain data. This suggests that the publisher within the experiment likely became overloaded, rendering it unresponsive to the testing tool. During data inspection, certain experiments exhibited an unexpectedly high number of samples. These instances were documented, revealing a discernible pattern of trailing zero values in the throughput measurements for subscribers. This issue was addressed by truncating any trailing zero values within the experiment's data. This process involves reversing the list of measurements, identifying the first non-zero value, removing all preceding elements, and subsequently reversing the list to its original order.

3.5.2 Filtering Out Experiments

All experiments are initially included during the processing of raw files into datasets. However, to properly ensure a good sampling of probabilities, experiments with less than 30 measurements are excluded from the final dataset. Subsequently, faulty experiments are systematically filtered from the generated dataset. Several recurring types of faults were identified.

In some instances, a sequence of 0 Mbps measurements was observed at the end of the experiment for the subscribers. To address this, each experiment was processed to remove all

trailing zeroes.

There were situations where the number of throughput measurements exceeded the expected number. For example, an experiment conducted for 600 seconds is anticipated to produce 600 measurements. However, in certain cases, a greater number of measurements were recorded. In such scenarios, all data from the expected endpoint onwards are truncated. For instance, if a 10-minute test yielded 900 measurements, data from the 600-second mark to 900 measurements are removed. The reliability of the data within the initial, “usable” portion is considered. Based on checks of a limited number of experiments, truncating the data at the expected endpoint was deemed appropriate. In some cases, values beyond this point diverged significantly, suggesting the concatenation of another experiment with the same identifier. In other cases, similar values persisted, indicating that some subscribers might have continued recording data post-experiment termination. In both scenarios, truncating the data from the expected endpoint onwards is considered a safe approach.

3.5.3 Transient and Outlier Analysis

To ensure the integrity of the data used for analysis, a two-stage process of transient and outlier analysis was developed. This process is critical for removing initialisation bias and handling anomalous data points that could otherwise skew the results.

Transient Period Analysis

The initial phase of any performance test can be subject to transient effects as the system stabilises. To ensure that our analysis is based on steady-state performance, we developed a systematic approach to identify and remove these periods. This methodology balances the need to eliminate initial non-stationarity with the imperative to preserve the maximum amount of valid experimental data.

For each dataset, a representative random sample of 100 experiments was subjected to graphical analysis. The time-series data for each experiment was plotted, and binned averages were overlaid to clarify performance trends (as illustrated in Figure 3.5). Each experiment was then classified based on the observed duration of its transient phase, categorised as resolving within 10%, 20%, 30%, 40%, or persisting beyond 40% of the test duration.

The final truncation percentage for a given dataset was determined by applying a **conservative threshold principle**. The goal was to select a single truncation value for the entire dataset that would confidently remove the transient periods from the vast majority of its con-

stituent experiments. For example, if analysis revealed that most transients resolved within the first 10% of the test duration, but a significant minority required up to 20%, the more conservative 20% threshold would be selected for all experiments in that dataset. This ensures methodological consistency and effectively filters initialisation bias across the entire dataset, even at the minor cost of discarding some early steady-state data from the fastest-stabilising experiments.

A unique behaviour was observed in certain datasets subjected to high network pressure, such as the “3Pi Machine Learning with Low Bandwidth Cap” campaign. In these cases, a significant minority of experiments displayed pronounced performance fluctuations beginning late in the experiment’s duration (e.g., after the 40% mark). Upon review, it was determined that these late-stage events are not representative of initial transient behaviour. Instead, they are hypothesised to be a critical performance characteristic of the system operating under sustained network pressure. These periods of instability likely represent the point at which system buffers are overwhelmed or other resource limits are reached. Classifying this behaviour as “transient” and removing it would therefore risk discarding valuable data needed to understand the system’s performance limits. In such cases, the most methodologically sound approach is to not apply a universal transient removal filter, ensuring that the full spectrum of performance behaviours is captured.

Outlier Analysis and its Critical Implications

The second stage of data cleaning involves the identification and handling of outliers. A standard approach, based on the transformation of data to standard scores (Z-scores), was initially considered. This method defines outliers based on a threshold of standard deviations from the mean (typically ± 2.5 for small samples and ± 4 for large samples).

However, the application of this method to our performance data proved to be unsuitable and highlighted a critical methodological challenge. The method consistently classified an unacceptably high percentage of data points as outliers (in some cases, over 90%). Especially since this was applied on a per-experiment basis, it resulted in the removal of nearly all data points from many experiments. This is a strong indication that the underlying assumption of a normal (or near-normal) distribution, upon which this outlier detection method is based, does not hold for our latency and throughput data. Network performance data is frequently characterised by non-normal, skewed, and multi-modal distributions, for which standard deviation-based outlier detection is inappropriate.

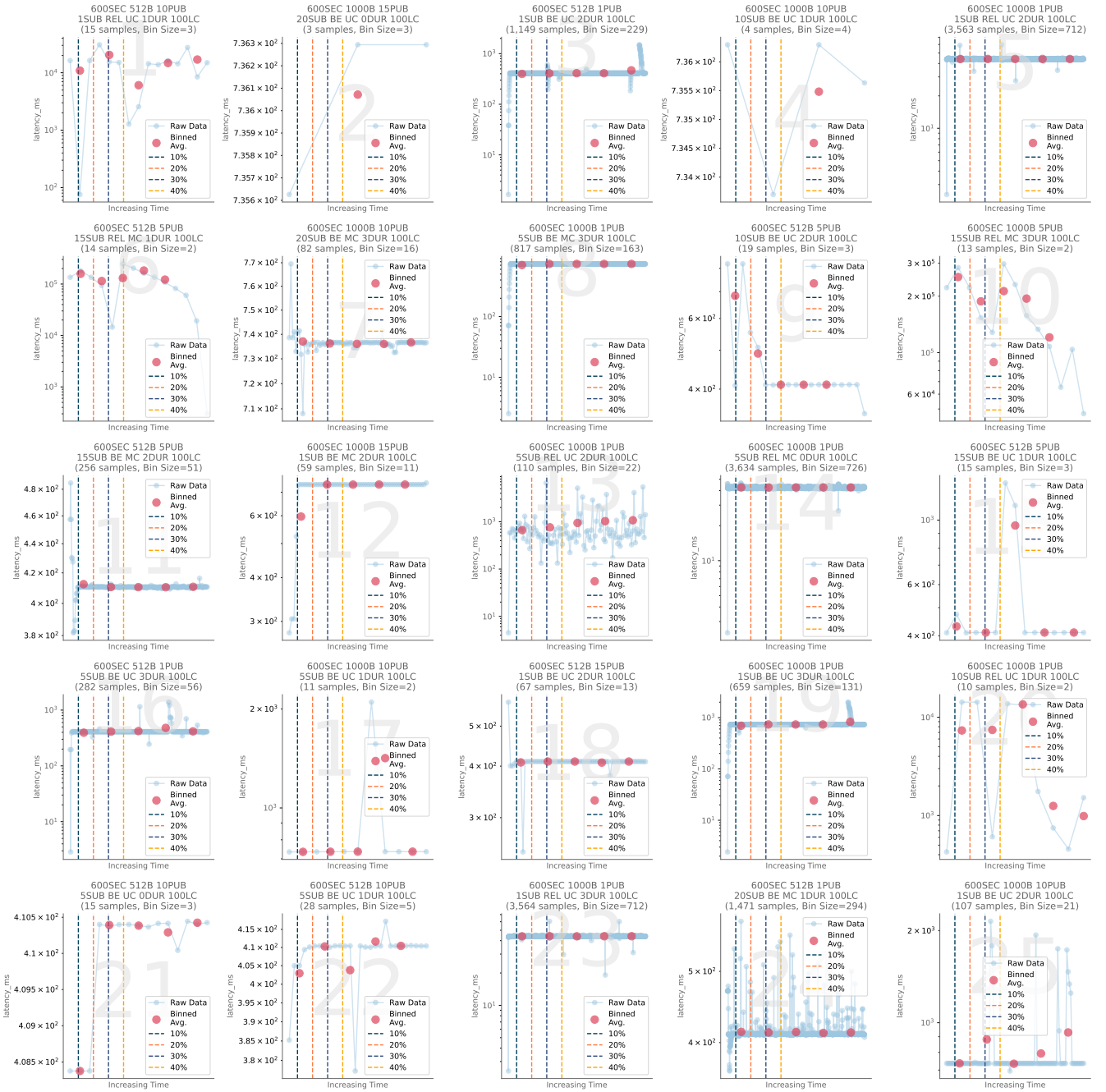


Figure 3.5: Example of Transient Analysis with Binned Averages for 25 Random Experiments

More importantly, a critical examination of what this method identifies as “outliers” revealed that these are often the most significant data points for this research. The high-latency events and sudden throughput drops are not mere statistical noise; they represent the system operating at its limits, under stress, or experiencing contention. To remove these data points would be to discard the very phenomena we seek to understand and predict. It would fundamentally undermine the goal of the analysis and lead to a sanitised but unrealistic view of DDS performance.

Therefore, a critical methodological decision was made: **no data points were removed**

from the ML datasets on the basis of being statistical outliers. This decision was made to ensure that the predictive models are trained on a complete and representative picture of the system’s behaviour, including its performance under stress. While this approach may increase the difficulty of the modelling task, it is essential for the development of robust and realistic performance prediction models. For the initial QoS capture datasets, where a very small percentage of extreme, anomalous values were observed (e.g., $<2\%$), a conservative filtering was applied. However, for the primary datasets used for predictive modelling, all data was retained. This approach ensures that the subsequent analysis is robust and that the predictive models are not biased by the unwarranted removal of critical performance data. The potential impact of these extreme values on model training is a key consideration in the discussion of the model-based prediction results in Chapter 5.

3.6 Datasets

This section describes the datasets used in our experiments. To reiterate, the datasets produced by the VMs are not included in this section, as the direct impact of DDS configuration parameters was confounded by the shared nature of the VM testbed. Therefore, the datasets are categorised according to the two physical testbeds established for this research: 3Pi and 5Pi. The majority of these datasets are the product of experimental campaigns conducted on these testbeds, which are detailed in Section 3.4. For each dataset, a concise description is provided, including the total number of experiments and the transient removal truncation percentages applied to each metric, as determined by the methodology outlined in Section 3.5.3.

3.6.1 3 Raspberry Pi Cluster

The 3Pi datasets originate from the experimental campaigns detailed in Section 3.4. There are five datasets in total, each corresponding to a specific experimental campaign, as enumerated in Table 3.15.

Dataset	Experiment Count
QoS Capture	1466
Multicast Exploration	50
ML No Bandwidth Cap	1537
ML Low Bandwidth Cap	339
ML High Bandwidth Cap	352
Total	3744

Table 3.15: Experiment Counts for 3Pi Datasets.

The campaigns from which these datasets were derived are as follows:

- QoS Capture.
- Multicast Exploration.
- ML, which is subdivided into:
 - Dataset without a bandwidth cap.
 - Dataset with a low bandwidth cap.
 - Dataset with a high bandwidth cap.

It should be noted that the ML datasets are created from a combination of other datasets. The truncation percentages applied to each dataset to account for transient effects, based on the methodology in Section 3.5.3, are displayed in Table 3.16.

Dataset	Latency Truncation (%)	Throughput Truncation (%)
QoS Capture	10	25
Multicast Exploration	10	10
ML No Bandwidth Cap	15	20
ML Low Bandwidth Cap	0	0
ML High Bandwidth Cap	10	0

Table 3.16: Transient Percentages for 3Pi Datasets.

ML Without a Bandwidth Cap

This dataset originates from a combination of several experimental campaigns and comprises 1537 experiments. To balance the removal of transient effects across this diverse set, the initial 15% of the latency data and 20% of the throughput data were truncated. As per the policy detailed in Section 3.5.3, no data points were removed as statistical outliers. Summary statistics of the numeric configuration parameter values for this dataset are shown in Table 3.17. To better visualise the configuration parameter space, a pairplot is provided in Figure 3.6.

Column	Mode	Min	Max	Median	Mean	Std. Dev.
Data Length (B)	32000	100	64000	32000	23330	25157
Publisher Count	10	1	75	10	21	22
Subscriber Count	1	1	75	10	24	24

Table 3.17: Summary Statistics for the Configuration Parameter Values for 3Pi ML Without a Bandwidth Cap.

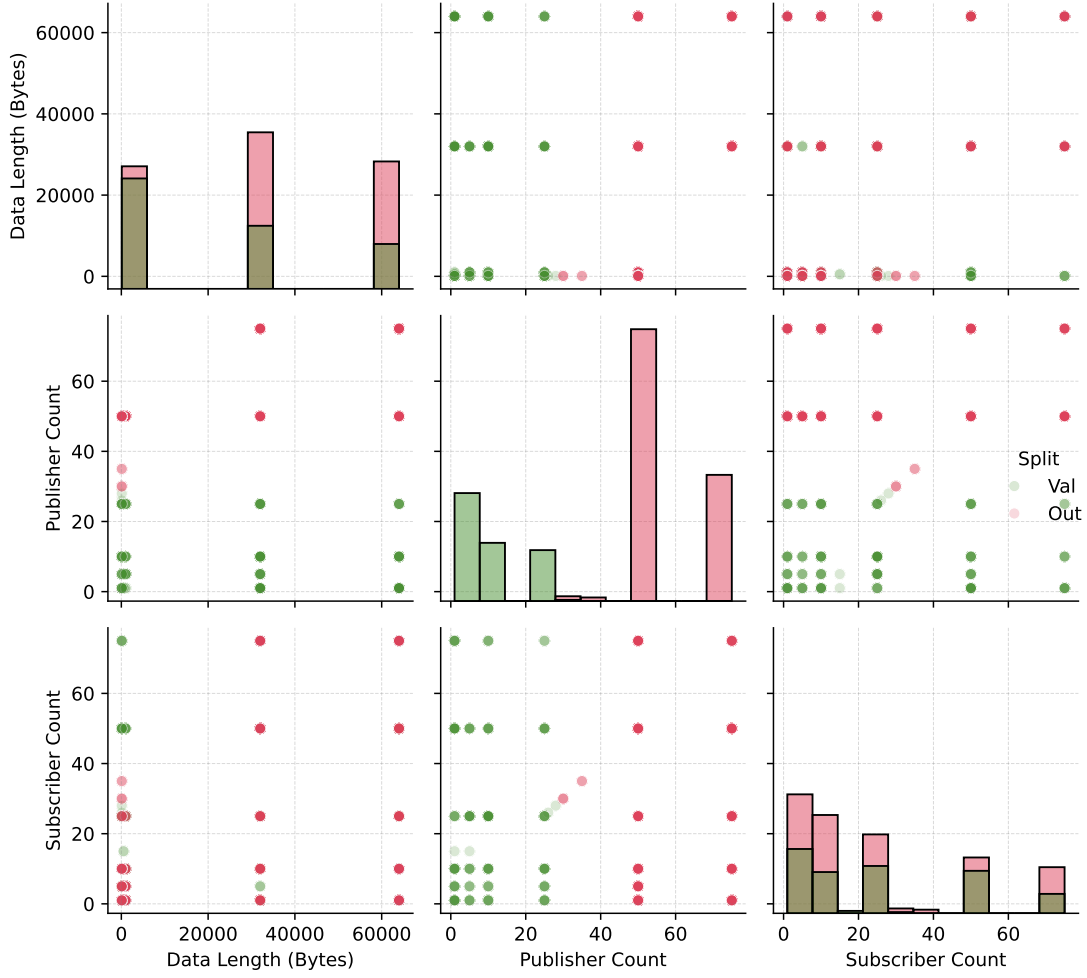


Figure 3.6: Pairplot of Configuration Parameters’ values for 3Pi ML Without a Bandwidth Cap split between “Val” (used for validating interpolation and shown in green) versus “Out” (used for validating extrapolation and shown in red).

ML with Low Bandwidth Cap (6 Mbps)

This dataset originates from the 3Pi ML campaign with a low bandwidth cap of 6 Mbps, containing 339 experiments. As detailed in Section 3.5.3, the performance instability observed late in these experiments was determined to be a critical performance characteristic under sustained network load, not initialisation bias. Therefore, no transient removal was applied to either the latency or throughput data for this dataset. Summary statistics of the numeric configuration parameter values for this dataset are presented in Table 3.18, and a pairplot is used to visualise the configuration parameter space in Figure 3.7.

Column	Mode	Min	Max	Median	Mean	Std. Dev.
Data Length (B)	512	512	1000	512	745.2035	244.1214
Publisher Count	1	1	20	5	4.8614	4.5236
Subscriber Count	1	1	20	10	8.6637	6.7174

Table 3.18: Configuration Parameters for 3Pi ML with Low Bandwidth Cap (6 Mbps).

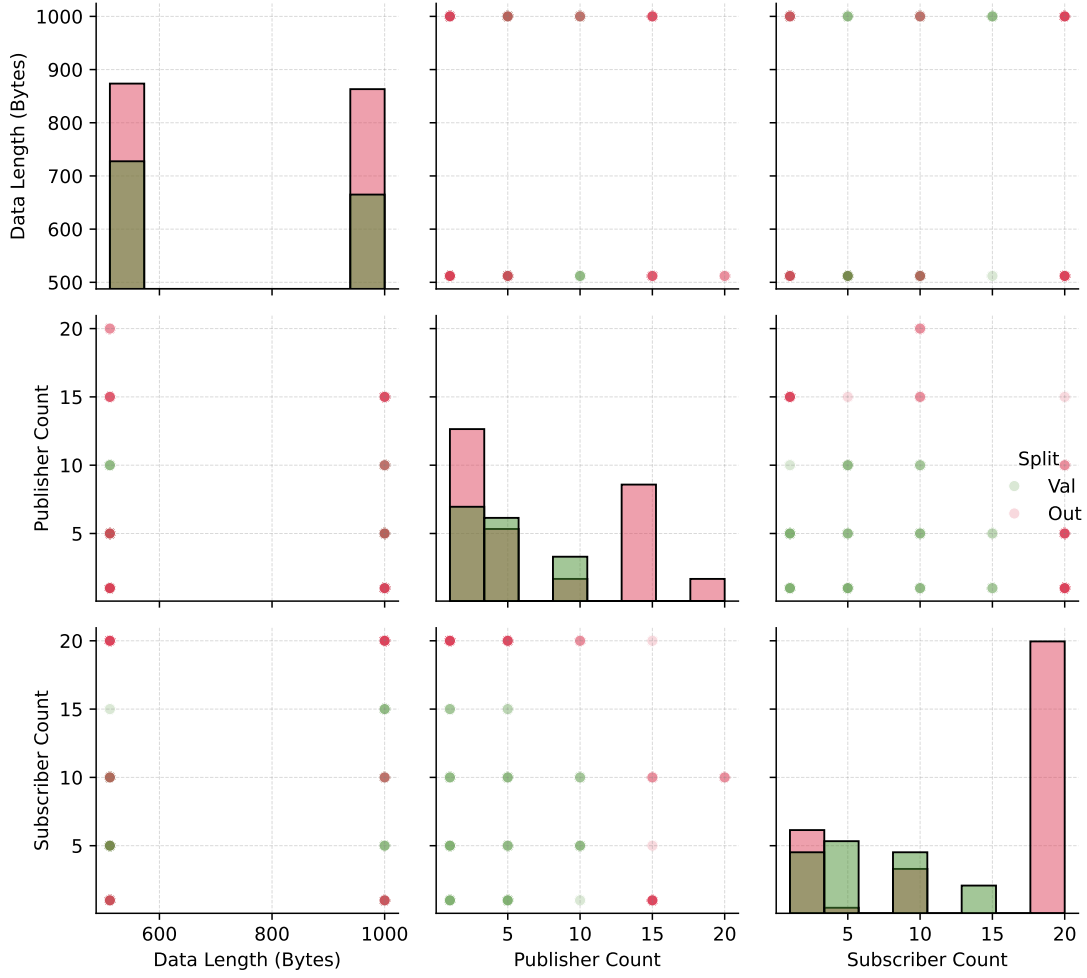


Figure 3.7: Pairplot of Configuration Parameters' values for 3Pi ML with Low Bandwidth Cap (6 Mbps) split between “Val” (used for validating interpolation and is shown in green) versus “Out” (used for validating extrapolation and is shown in red).

ML with High Bandwidth Cap (8 Mbps)

This dataset originates from the 3Pi ML campaign with a high bandwidth cap of 8 Mbps and contains 352 experiments. Transient analysis revealed that latency measurements stabilised consistently within the first 10%, which was therefore truncated. For throughput, late-stage instability was deemed a key performance characteristic under this specific load, so no transient removal was applied. Summary statistics of the numeric configuration parameter values for this dataset are shown in Table 3.19, with a corresponding pairplot visualisation in Figure 3.8.

Column	Mode	Min	Max	Median	Mean	Std. Dev.
Data Length (B)	512	512	1000	512	739.3636	243.7787
Publisher Count	1	1	20	5	5.8153	5.2333
Subscriber Count	1	1	20	10	8.9574	7.1493

Table 3.19: Configuration Parameters for 3Pi ML with High Bandwidth Cap (8 Mbps).

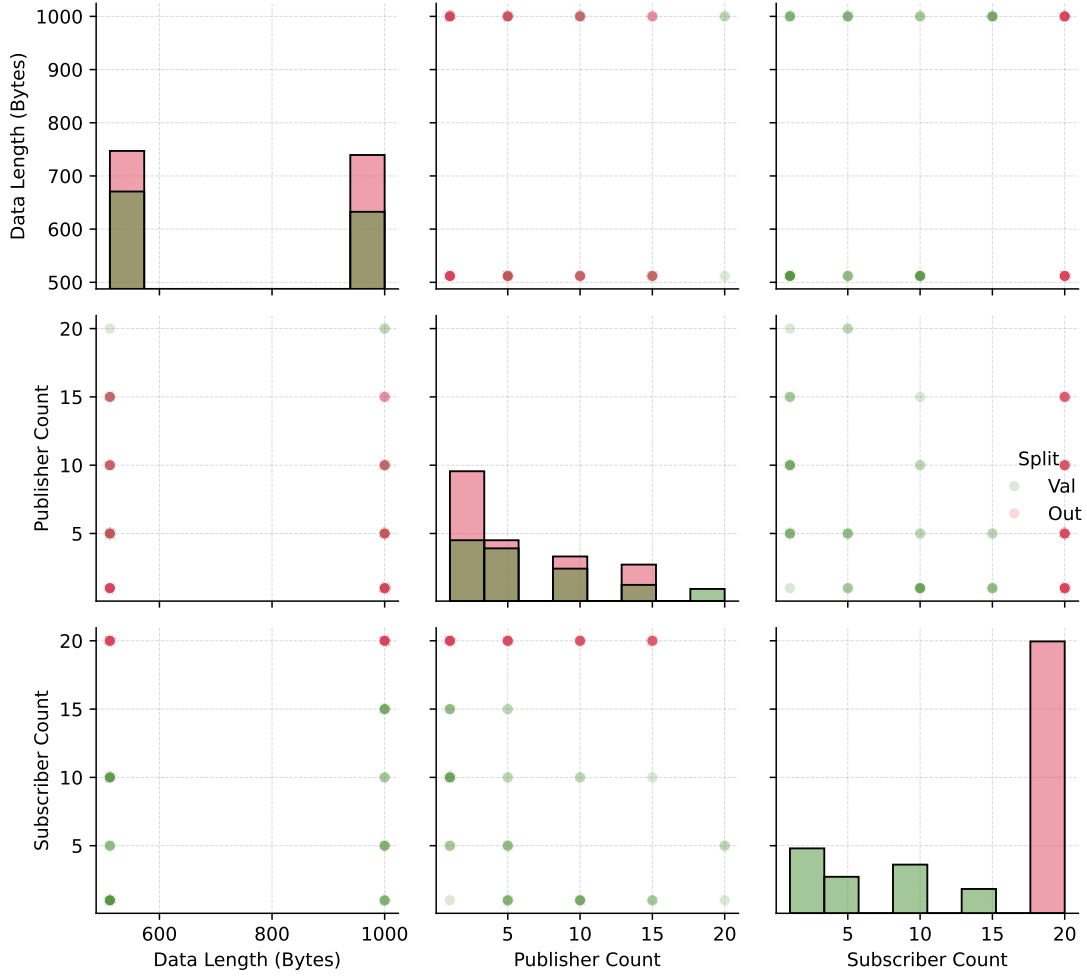


Figure 3.8: Pairplot of Configuration Parameters for 3Pi ML with High Bandwidth Cap (8 Mbps) split between “Val” (used for validating interpolation and is shown in green) versus “Out” (used for validating extrapolation and is shown in red).

3.6.2 5 Raspberry Pi Cluster

For the 5Pi testbed, we generated several datasets by varying the allocation of participants per node. These datasets use the notation $xPyS$, where x denotes the number of machines allocated solely to publishers, and y denotes the number of machines allocated solely to subscribers. In cases where no notation is used, it indicates a 2P2S configuration where machines are allocated to both publishers and subscribers. A summary of the datasets, including the number of experiments and truncation percentages for latency and throughput, is provided in Table 3.20.

ML Without a Bandwidth Cap

This dataset originates from a combination of several 5Pi campaigns and contains 549 experiments. A conservative 10% of the latency data and 20% of the throughput data were truncated. No outlier removal was performed. Summary statistics of the numeric configuration parameter values for this dataset are shown in Table 3.21, and a pairplot visualisation is provided in Figure

Campaign	Experiment Count	Latency Truncation (%)	Throughput Truncation (%)
QoS Capture (2P2S)	110	10	20
QoS Capture (3P1S)	107	10	10
QoS Capture (1P3S)	179	10	20
Multicast Exploration (1P3S)	62	10	10
Multicast Exploration (2P2S)	49	10	10

Table 3.20: Summary of experiment counts and truncation percentages for latency and throughput in the 5Pi Quality of Service (QoS) Capture and Multicast Exploration campaigns.

3.9.

Column	Mode	Min	Max	Median	Mean	Std. Dev.
Data Length (B)	100	32	128000	100	16256	33277
Publisher Count	1	1	28	5	6	7
Subscriber Count	1	1	75	9	10	9

Table 3.21: Configuration Parameters for 5Pi ML Without a Bandwidth Cap.

ML with Low Bandwidth Cap (24 Mbps)

This dataset originates from the 5Pi ML campaign with a low bandwidth cap of 24 Mbps and contains 629 experiments. For latency, a 10% truncation was applied. For throughput, late-stage instability was determined to be a key performance characteristic, so no transient removal was applied. Summary statistics of the numeric configuration parameter values for this dataset are shown in Table 3.22, with a corresponding pairplot visualisation in Figure 3.10.

Column	Mode	Min	Max	Median	Mean	Std. Dev.
Data Length (B)	512	512	1000	512	747.08	244.03
Publisher Count	1	1	20	10	8.63	6.46
Subscriber Count	1	1	20	10	9.08	6.69

Table 3.22: Configuration Parameters for 5Pi ML with Low Bandwidth Cap (24 Mbps).

ML with High Bandwidth Cap (71 Mbps)

This dataset originates from the 5Pi ML campaign with a high bandwidth cap of 71 Mbps and contains 610 experiments. Applying the conservative threshold principle, a 20% truncation was applied to latency data and a 30% truncation to throughput data to account for transients observed across the constituent experiments. Summary statistics of the numeric configura-

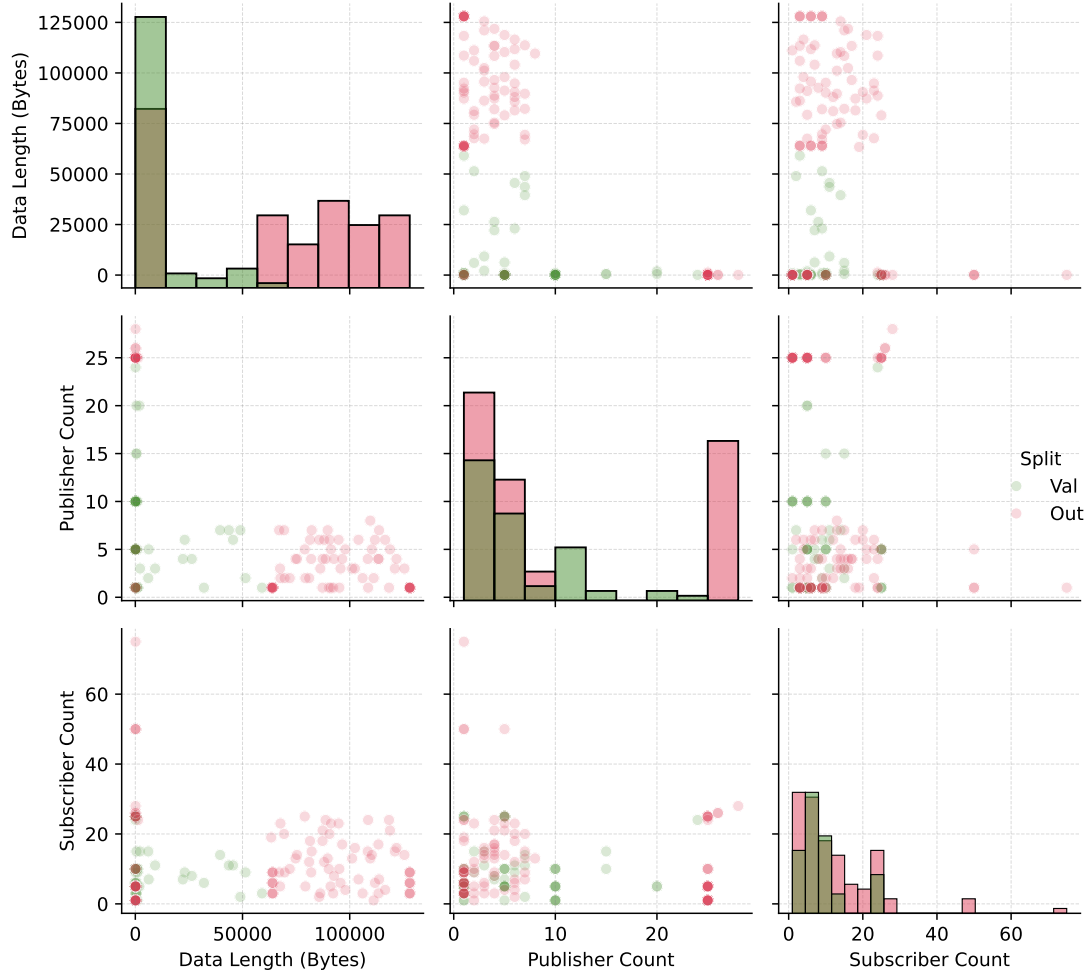


Figure 3.9: Pairplot of Configuration Parameters for 5Pi ML Without a Bandwidth Cap split between “Val” (used for validating interpolation and is shown in green) versus “Out” (used for validating extrapolation and is shown in red).

tion parameter values for this dataset are shown in Table 3.23, and a pairplot visualisation is provided in Figure 3.11.

Column	Mode	Min	Max	Median	Mean	Std. Dev.
Data Length (B)	512	512	1000	512	746.40	244.01
Publisher Count	1	1	20	10	9.81	7.01
Subscriber Count	5	1	20	10	10.05	6.78

Table 3.23: Configuration Parameters for 5Pi ML with High Bandwidth Cap (71 Mbps).

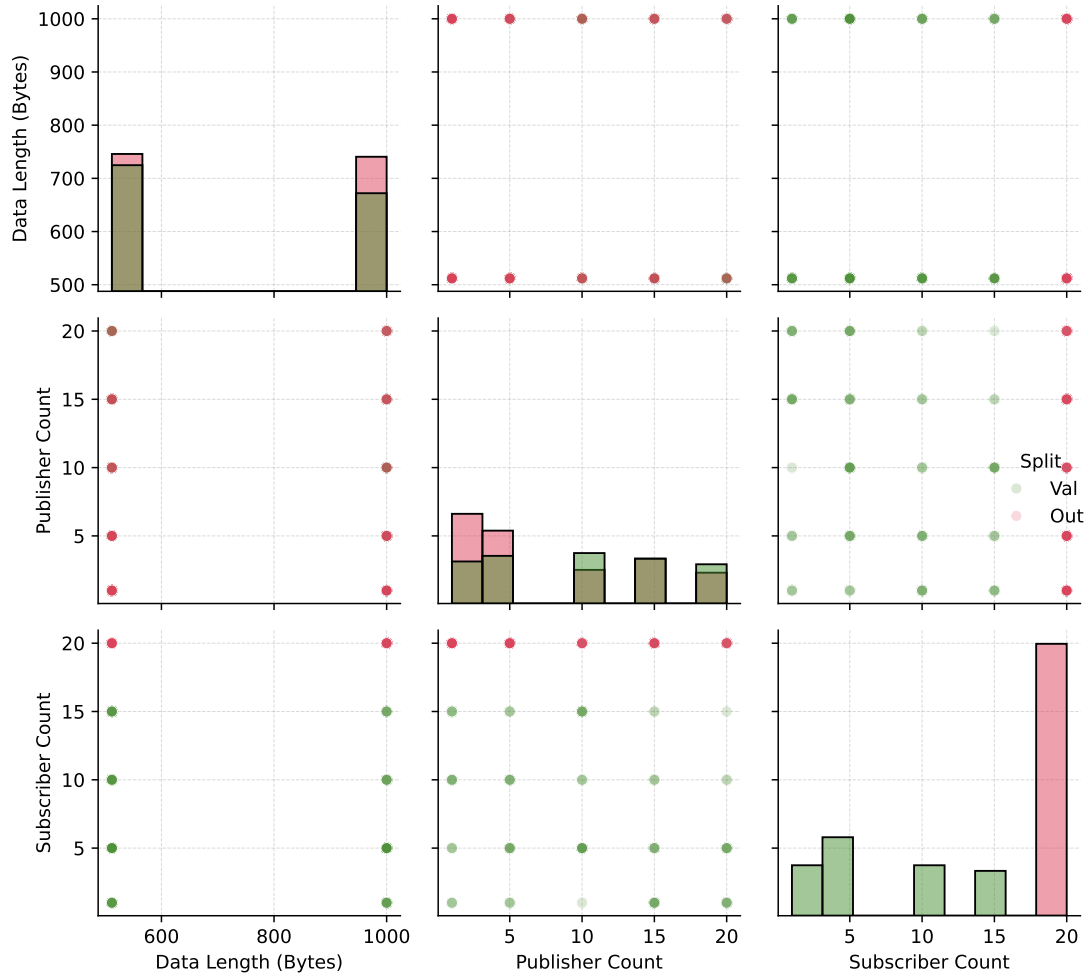


Figure 3.10: Pairplot of Configuration Parameters for 5Pi ML with Low Bandwidth Cap (24 Mbps) split between “Val” (used for validating interpolation and is shown in green) versus “Out” (used for validating extrapolation and is shown in red).

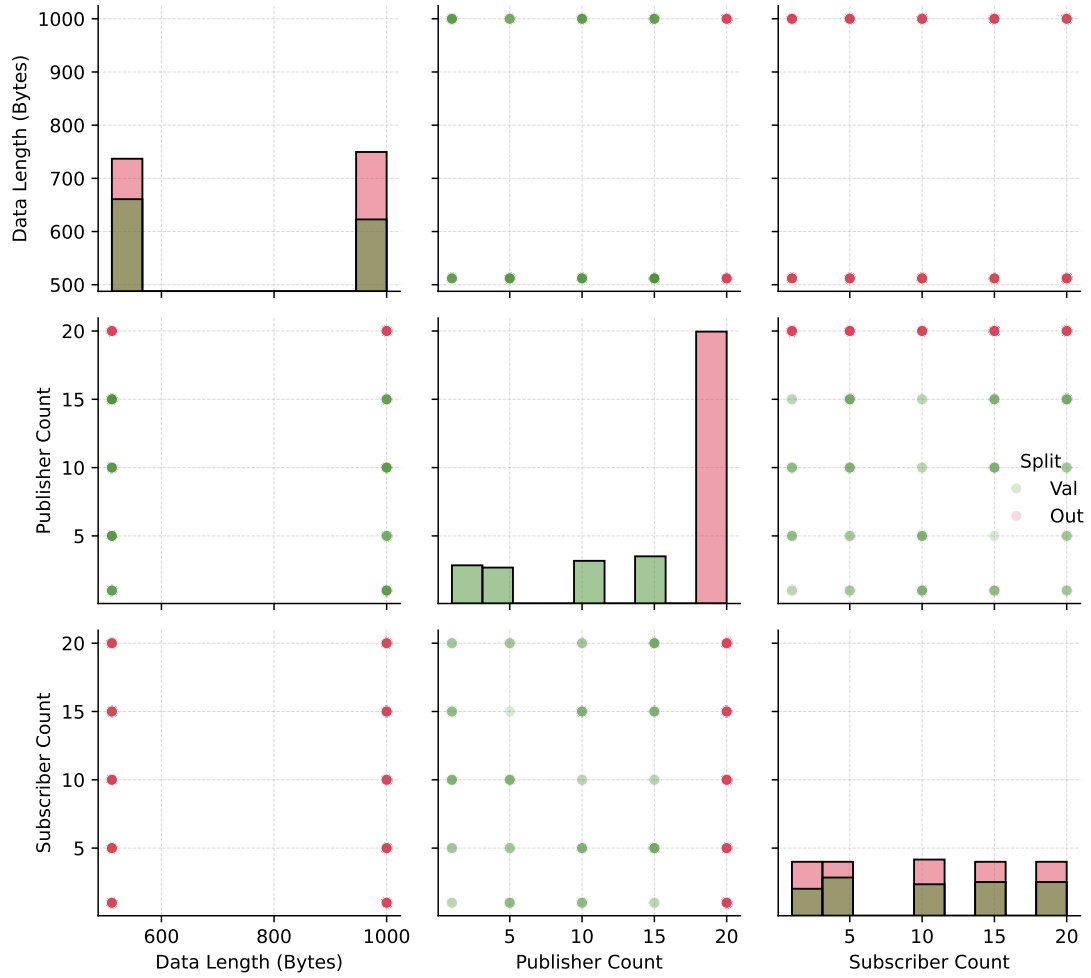


Figure 3.11: Pairplot of Configuration Parameters for 5Pi ML with High Bandwidth Cap (71 Mbps) split between “Val” (used for validating interpolation and is shown in green) versus “Out” (used for validating extrapolation and is shown in red).

Chapter 4

Experimental Performance Evaluation of Data Distribution Service

4.1 Introduction

This chapter presents a comprehensive experimental performance evaluation of DDS. The primary objective is to systematically investigate the impact of various configuration parameters, including QoS policies (e.g. reliability and durability), and network-related factors (e.g. multicast communication, message size, and network noise), on the performance of DDS in terms of latency and throughput. This is implemented via RTI's *Connex DDS* implementation, specifically through the use of RTI's *Perftest* benchmarking tool. Delays in data delivery or bottlenecks in throughput can have severe consequences in applications such as autonomous driving, air traffic control, and industrial automation. Evaluating the performance of DDS will provide a deep understanding of how different settings affect the real-time communication capabilities in various scenarios, and may reveal counter-intuitive performance characteristics. Therefore, a thorough performance evaluation is essential to ensure that DDS can meet the stringent requirements of these systems.

This analysis is conducted across three distinct testbeds: a virtualised environment and two physical RPi clusters of varying sizes (3Pi and 5Pi). Latency and throughput measurements were collected for a wide range of configurations, varying QoS policies, message sizes, and network conditions as detailed in Chapter 3. Majority of the analysis is done via the use of eCDFs which show the total percentage of measurements that are less than or equal to a specific value ranging from 0 to 1 (0% to 100%). For a given point (x, y) on the eCDF, y represents the proportion of measurements that are less than or equal to x . These results are

analysed to identify trends, anomalies, and the influence of data length, publisher/subscriber count, reliability, use of multicast communication, and durability on latency and throughput performance.

4.1.1 Contributions

The main contributions of this chapter are threefold.

1. Provides an objective view of the results when assessing the performance evaluation of DDS under a wide range of configurations, offering insights into its behaviour and revealing unexpected performance trends.
2. Identifies the most influential QoS parameters and quantifies their impact on latency and throughput.
3. The experimental data collected in this study serves as the foundation for the subsequent chapter on model-based performance prediction as well as for any parties interested in carrying out further analysis and prediction of such data.

4.1.2 Structure of the Chapter

This chapter is organised as follows. Section 4.2 evaluates the performance of DDS in virtualised environments. Section 4.3 discusses the impact of multicast on DDS performance, including its often counter-intuitive behaviour across different environments. Section 4.4 investigates the effect of data length on latency and throughput. Section 4.5 examines the influence of network noise on DDS performance. Section 4.6 analyses the causes and implications of 0 Mbps throughput measurements which were observed at a high frequency in the experiments. Section 4.7 discusses the limitations of the experimental setup used in this study. Finally, Section 4.8 concludes the chapter and outlines future work.

4.2 Virtualised Environments

Virtualisation is a common practice in modern computing infrastructure, and understanding its impact on DDS performance is crucial for real-world deployments. The use of VMs introduces an additional layer of complexity to network communication, which can potentially affect the performance of real-time systems like DDS. This section provides insight into the performance characteristics of DDS when deployed in virtualised settings, highlighting the challenges and considerations that arise in such environments.

A series of experimental campaigns were designed to systematically evaluate the performance of DDS in the VM testbed. These campaigns aimed to explore various aspects of DDS performance, including the effects of MC communication, resilience to DDoS attacks, the impact of malicious subscribers, and the influence of network noise. Below is an overview of each experimental campaign:

- **Subsection 4.2.1 - Base Case Campaign:** Establishes a baseline for DDS performance, comparing UC and MC communication under standard conditions.
- **Subsection 4.2.2 - Exploration Campaign:** Investigates the impact of MC on latency and throughput under different conditions.
- **Subsection 4.2.3 - DDoS Campaign:** Evaluates the resilience of DDS against simulated DDoS attacks.
- **Subsection 4.2.4 - Subscriber Attack Campaign:** Examines the effect of malicious subscribers on the performance of DDS, simulating an attack scenario where subscribers consume resources not intended for them.
- **Subsection 4.2.5 - Confirmation of Variation Campaign:** Conducting additional experiments to confirm the presence of variation in the measurements.
- **Subsection 4.2.6 - Single versus Multiple Machine Campaign:** Compares the performance of DDS when deployed on a single machine versus multiple machines.

4.2.1 Base Case

Unicast VS Multicast Latencies

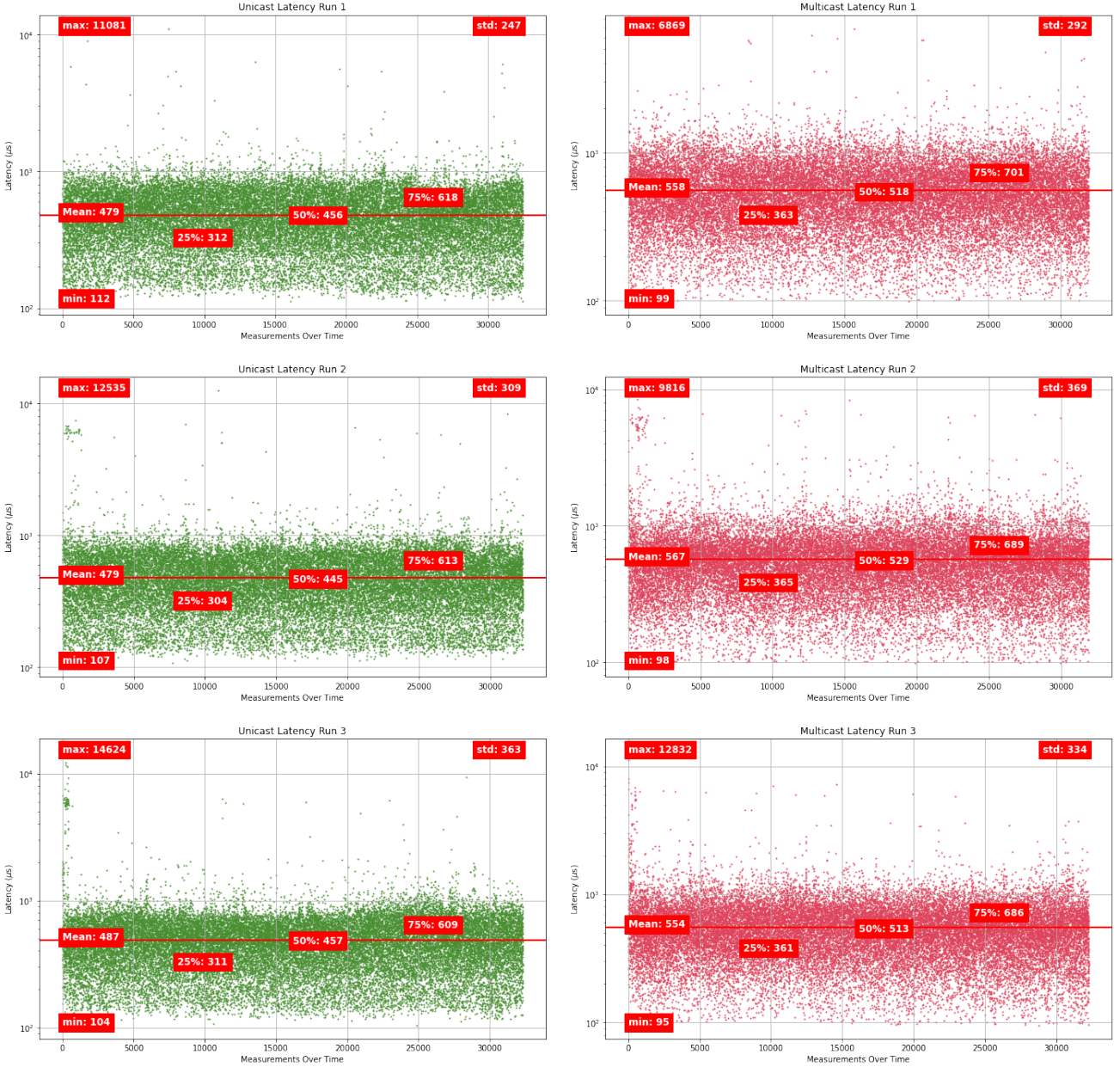


Figure 4.1: Latency measurements for the base case campaign, comparing UC and MC communication over time across 3 runs.

The “Base Case” experimental campaign established an initial understanding of DDS performance within virtualised environments. Figure 4.1 shows the latency measurements over time for both UC and MC communication across 3 runs and the throughput eCDFs are seen in Figure 4.2 for the same conditions. Latency measurements revealed a range of 104 to 14624 μs for UC communication, with a mean of 483 μs . Multicast communication exhibited a latency range of 95 to 12832 μs , with a mean of 560 μs . The presence of maximum latency values at the commencement of experiments suggests a warm-up period for the system. Median latency

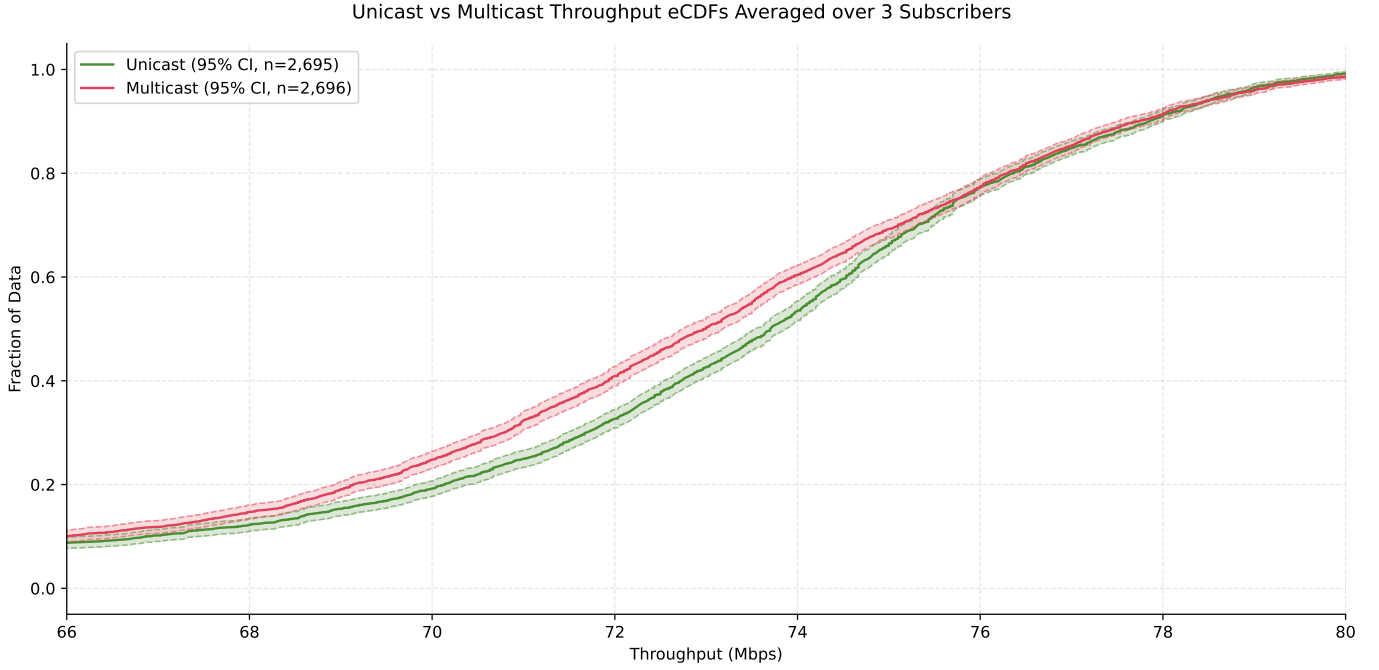


Figure 4.2: eCDF of throughput for the base case campaign, comparing UC and MC communication across all runs across each individual subscriber.

figures were $453 \mu\text{s}$ for UC and $520 \mu\text{s}$ for multicast. In terms of throughput, both UC and multicast demonstrated comparable performance, ranging between 40 to 80 Mbps across all 3 experimental runs. These findings indicate that, under standard conditions, both communication methods deliver similar throughput, while multicast exhibits slightly higher mean and median latency compared to UC. The initial high latency values underscore the importance of allowing systems to stabilise before commencing performance-critical measurements. This “Base Case” provides a source of comparison for subsequent experimental campaigns.

4.2.2 Multicast Exploration

Whilst not specified within the DDS standard, the use of MC is an impactful factor of DDS performance that can significantly improve the efficiency of data distribution to multiple subscribers. However, its performance can be affected by various factors, such as network configuration and the number of subscribers. This campaign investigates the performance of DDS with MC enabled, exploring its impact on latency and throughput under different conditions and further details are provided in Subsection 3.4.3. The results of the MC exploration campaign are presented in Figures 4.3, 4.4, 4.5, and 4.6.

The latency box plots in Figure 4.3 show an almost-linear increase in latency with the number of participants for both UC and MC communication. The latency ranges from 0.1 to 0.4 ms for 1 publisher and 1 subscriber and from 0 to 700 ms for 75 publishers and 75

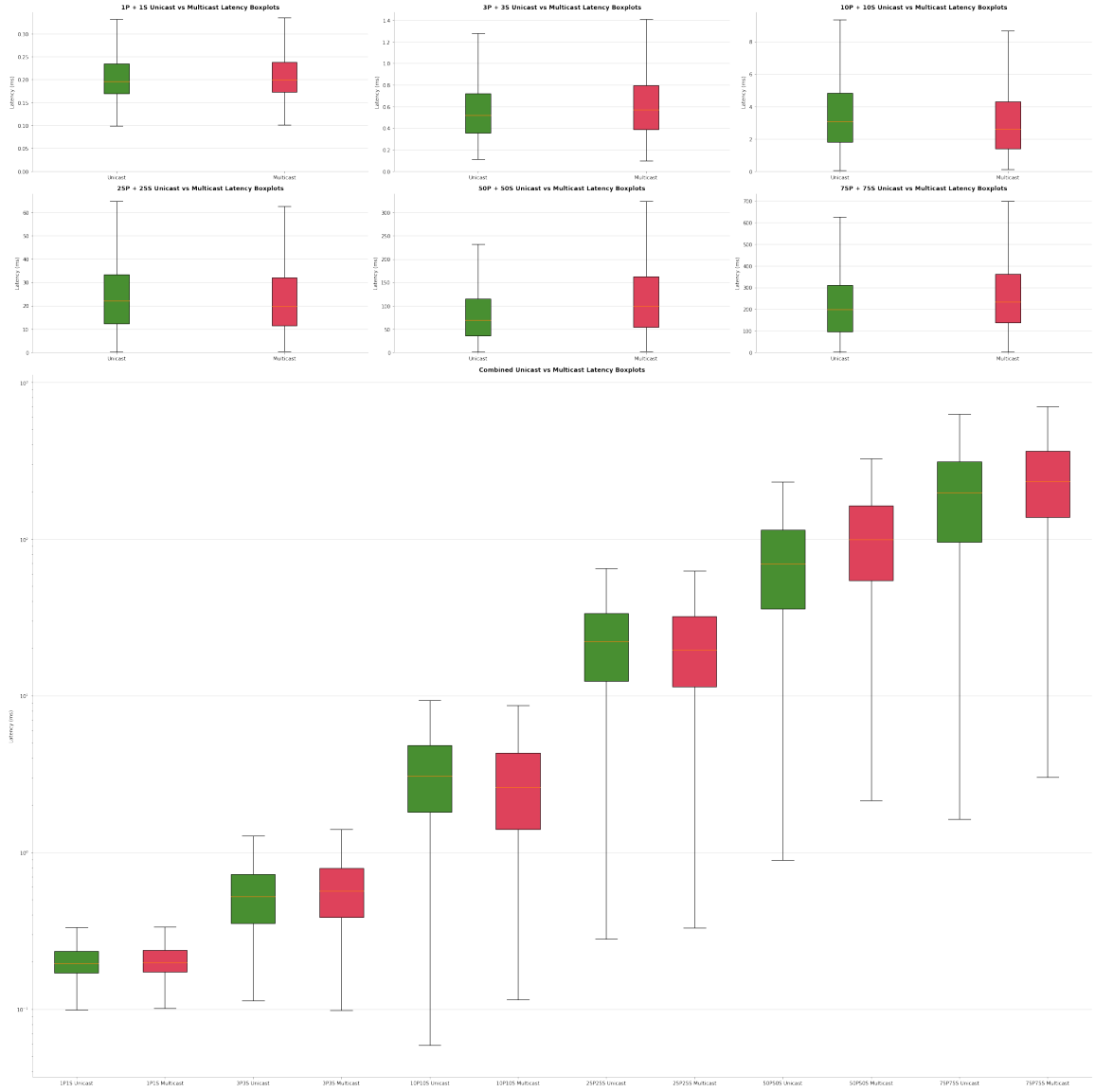


Figure 4.3: Box plots comparing latency for UC in green and MC in red communication across different configurations in the MC exploration campaign.

subscribers.

Figure 4.4 shows that UC outperforms MC for 3, 50, and 75 publishers and subscribers, while MC outperforms UC for 10 publishers and 10 subscribers. Similar performance is observed for 3 and 25 publishers and subscribers for both UC and MC. In the case of 3 publishers and 3 subscribers there is an evident peak at 500 μ s.

A linear increase in throughput with the number of participants is not observed in Figure 4.5. The highest throughput is seen for 10 publishers and 10 subscribers, especially for MC communication, which reaches up to 340 Mbps. The throughput ranges from 80 to 100 Mbps for 1 publisher and 1 subscriber, and from 220 to 340 Mbps for 10 publishers and subscribers. The following observations are seen in Figure 4.6. Similar performance is seen for 3 publishers and 3 subscribers for both UC and MC with a peak around 190 Mbps and throughput ranging from

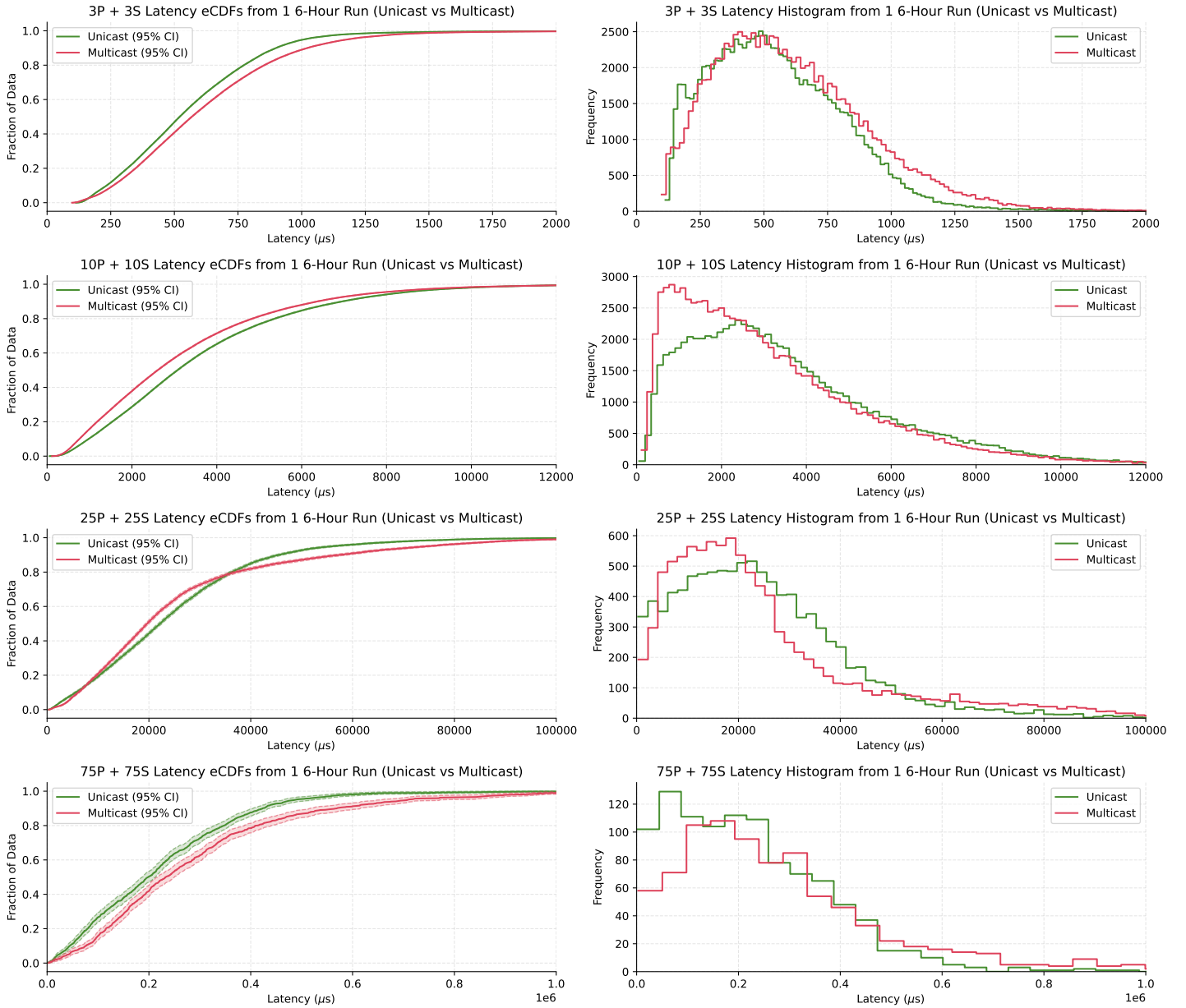


Figure 4.4: eCDFs (left column) and histograms (right column) of latency for UC (green) and MC (red) communication across 3, 10, 25, and 75 publishers/subscribers in the MC exploration campaign. 95% confidence intervals included.

5 to 225 Mbps. For 10 publishers and subscribers, MC outperforms UC with a peak around 290 Mbps (versus 250 Mbps for UC). The eCDF also demonstrates stochastic ordering (a visual separation of both lines) in favour of MC. MC is also more performant for 25 publishers and 25 subscribers with a peak around 220 Mbps (versus 200 Mbps for UC). The eCDF shows a crossover around 185 Mbps and then MC outperforms UC for the rest of the distribution. In the case of 50 publishers and 50 subscribers, MC again outperforms UC with a peak around 180 Mbps (versus 175 Mbps for UC) with clear stochastic ordering in favour of MC. A switch in performance is observed for 75 publishers and 75 subscribers, where UC outperforms MC with a peak around 180 Mbps (versus 165 Mbps for MC) with a clear stochastic ordering in

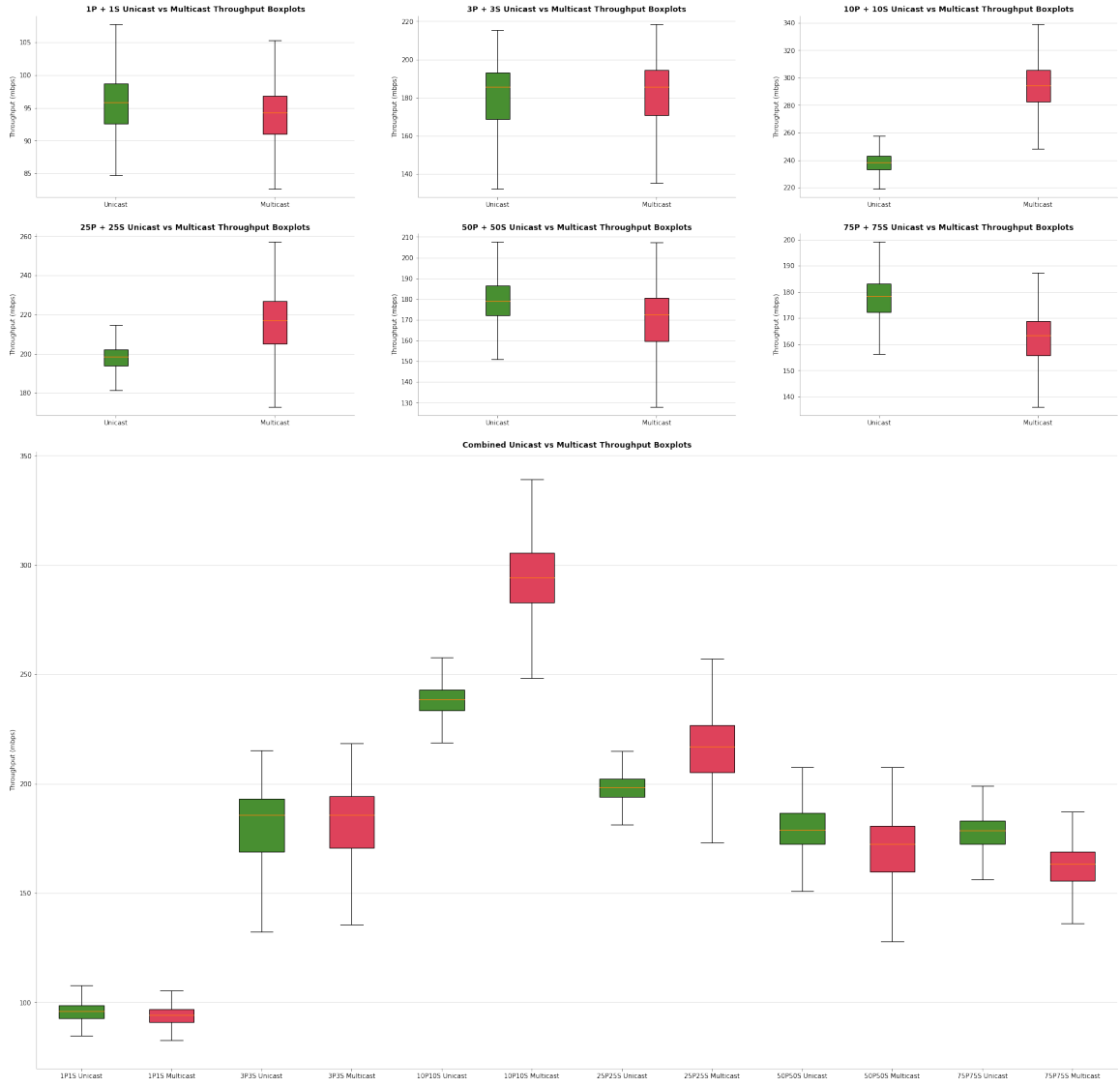


Figure 4.5: Box plots comparing throughput for UC in green and MC in red communication across different configurations in the MC exploration campaign.

favour of UC.

In summary, an almost-linear increase in latency is observed with the number of participants for both communication types. UC outperforms MC for 3, 50, and 75 publishers and subscribers, MC outperforms UC for 10 publishers and 10 subscribers, and similar performance is observed for 3 and 25 publishers and subscribers in terms of latency. This almost-linear increase in latency is not observed in throughput, where no clear trend is seen with the number of participants. The highest throughput is observed for 10 publishers and 10 subscribers, especially for MC communication, which reaches up to 340 Mbps. UC outperforms MC for 3 and 75 publishers and subscribers, while MC outperforms UC for 10, 25, and 50 publishers and subscribers in terms of throughput.

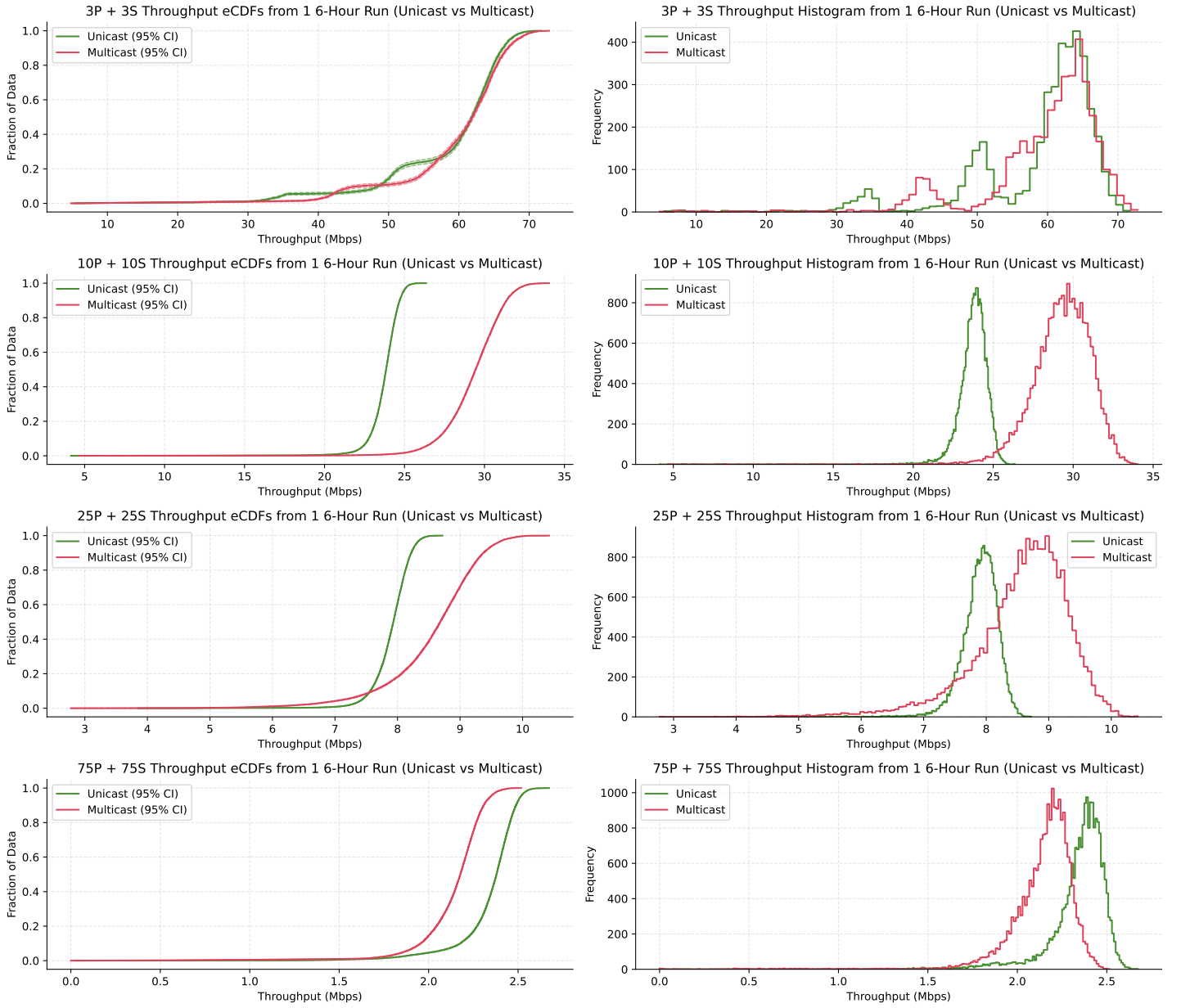


Figure 4.6: eCDFs (left column) and histograms (right column) of throughput for UC (green) and MC (red) communication across 3, 10, 25, and 75 publishers/subscribers in the MC exploration campaign. 95% confidence intervals included.

4.2.3 Distributed Denial of Service

Data Length	UC Mean Difference (ms)	MC Mean Difference (ms)
500 B	~24	~27
1 KB	~29	~30
16 KB	~31	~25
64 KB	~30	~14
128 KB	~24	~33

Table 4.1: Mean differences in latency for UC and MC communication during the DDoS campaign across different data lengths.

DDoS attacks constitute a formidable challenge to the operational integrity of network

services. This investigation evaluates the resilience of the DDS against such DDoS attacks. It specifically examines the system’s capacity to sustain effective communication in the presence of malicious network traffic. Details of this experimental campaign can be found in Subsection 3.4.3.

The increases in mean latency for UC and MC communication during the DDoS campaign are detailed in Table 4.1. These increases confirm that DDoS attacks elevate latency for both UC and MC communication, as anticipated. However, no periods of communication failure are observed, indicating that the DDS maintains functionality without significant disruption during a DDoS attack. For a 16 KB payload, the ordering of UC and MC latency reverses, with UC exhibiting a mean latency of approximately 26 ms and MC approximately 29 ms under normal conditions. During a DDoS attack, the UC mean latency rises to approximately 57 ms, while the MC mean latency increases to approximately 54 ms. This suggests that DDoS attacks substantially affect the latency of both communication methods, though MC maintains relatively stable communication. For a 64 KB payload, a similar reversal in latency ordering occurs, with UC mean latency at approximately 26 ms and MC at approximately 29 ms during normal operation. Under a DDoS attack, the UC mean latency increases to approximately 56 ms, while the MC mean latency rises to approximately 43 ms. For a 128 KB payload, the UC mean latency increases to approximately 50 ms during a DDoS attack, while the MC mean latency rises to approximately 52 ms. Notably, the latency increase at 128 KB is less pronounced than at 1 KB, suggesting that DDS can handle larger data payloads with minimal communication disruption. A linear relationship between message size and the difference in UC or MC mean latency differences is not observed. For UC communication, a 128 KB payload, the largest tested, is as ineffective at increasing latency as a 500 B payload, the smallest tested. For MC communication, a 64 KB payload is the least effective at increasing latency, performing better than both 500 B and 128 KB payloads, while the 128 KB payload is the most effective, as expected.

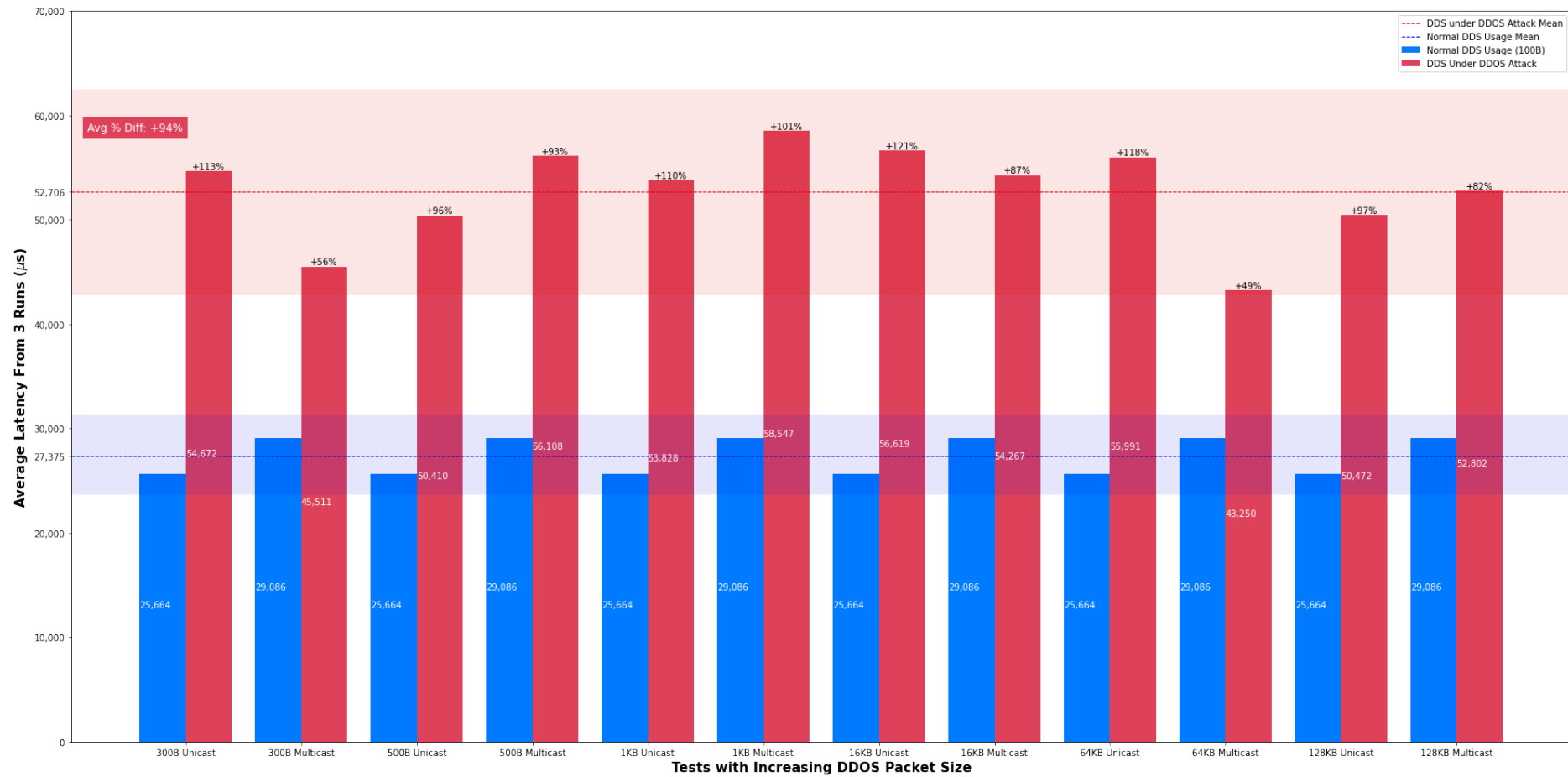


Figure 4.7: Average latency for the DDoS campaign with 25 publishers and 25 subscribers, comparing UC and MC communication across different data lengths.

Figure 4.7 illustrates the average latency observed during the DDoS campaign, involving 25 publishers and 25 subscribers, and compares UC and MC communication across varying DDoS data payload sizes. As indicated in Table 4.2, DDoS attacks increase latency by an average of 94%, nearly doubling the baseline values. For UC communication, the smallest impact on latency occurs with a 500 B payload, exhibiting a 96% increase, though the 128 KB payload is marginally higher by 1%. The greatest impact on UC latency is observed with a 16 KB payload, which results in a 121% increase. For MC communication, the least impact on latency is seen with a 64 KB payload, showing a 49% increase. The most significant impact on MC latency occurs with a 1 KB payload, resulting in a 101% increase. These findings suggest that, for maximising the effectiveness of a DDoS attack, a 16 KB payload should be used with UC communication instead of the expected larger 128 KB payload. Alternatively, if employing MC communication, a 1 KB payload is most effective, again contrary to the expectation of using the largest tested payload. When unrestricted in choice, a 16 KB payload with UC communication yields the highest latency impact.

Data Length	UC Mean Difference (%)	MC Mean Difference (%)
300 B	113	56
500 B	96	93
1 KB	110	101
16 KB	121	87
64 KB	118	49
128 KB	97	82
Average	109	78
Min	96	49
Max	121	101

Table 4.2: Mean differences in latency for UC and MC communication during the DDoS campaign across different data lengths, expressed as percentages.

4.2.4 Malicious Subscriber Attack

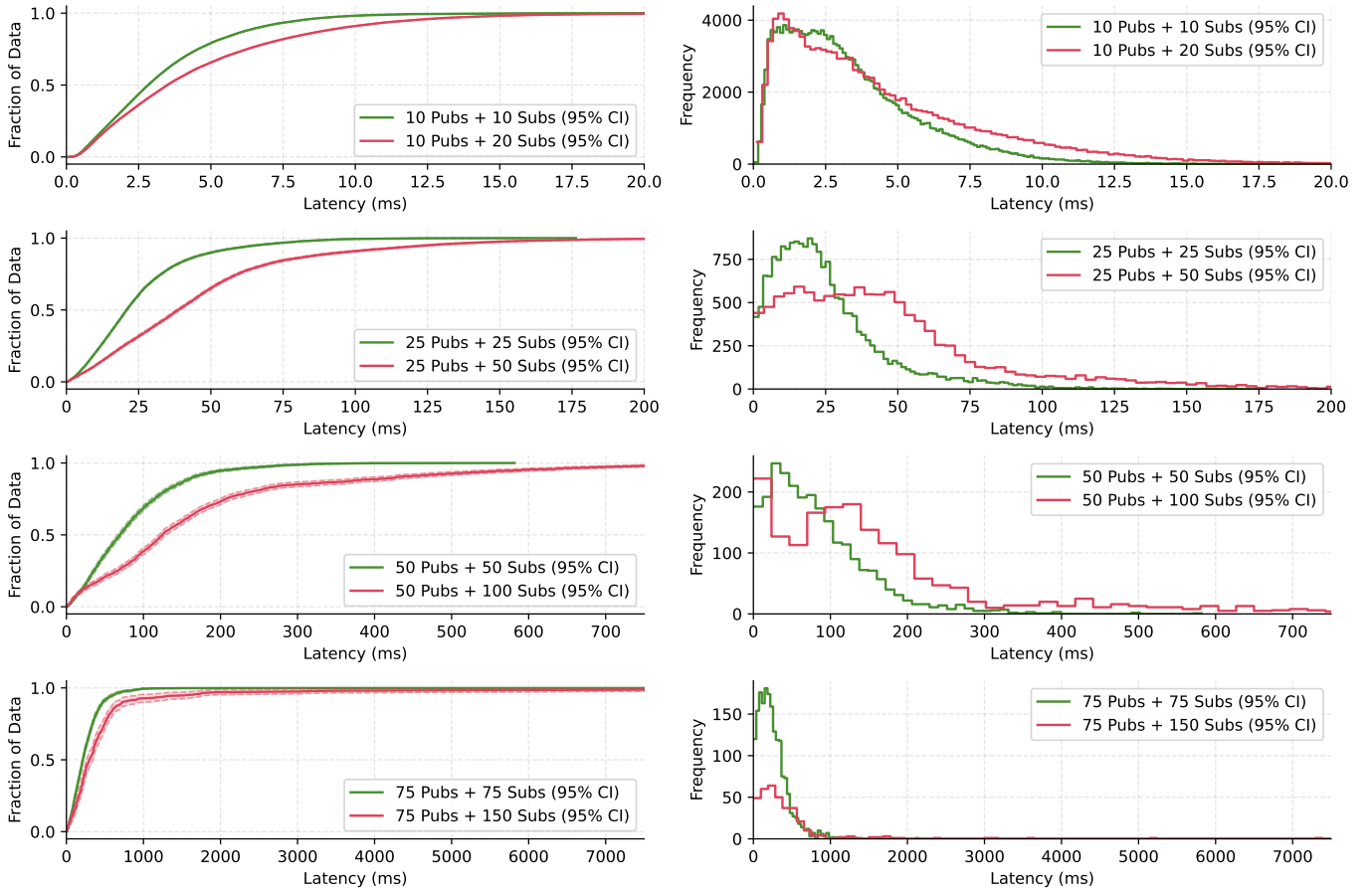


Figure 4.8: eCDFs with 95% confidence intervals (left column) and histograms (right column) of latency (ms) for normal (green) and malicious (red) cases with 10, 25, 50, and 75 publishers.

In addition to external attacks, DDS systems can also be vulnerable to attacks from within, such as a malicious subscriber attempting to eavesdrop on communication. This campaign investigates the impact of a malicious subscriber on the performance of DDS. Details of the experiments can be found in Subsection 3.4.3.

Analysis of the eCDFs in Figure 4.8 reveals that additional malicious subscribers will increase the latency with clear stochastic ordering observed for all cases. Examination of the Probability Density Function (PDF) shows that they resemble skewed normal distributions which suggests a common behavioural pattern across the distributions. As the number of subscribers increases, the peaks of these distributions gradually shift towards higher latency values. This indicates a consistent and predictable degradation in performance with increased subscriber load. A distinct stochastic ordering was observed across different subscriber counts, unequivocally showing that a higher number of subscribers consistently led to elevated latency.

Analysis of the eCDF in Figure 4.9 show similar change in performance related to throughput. Adding malicious subscribers results in a decrease in throughput, easily identified as

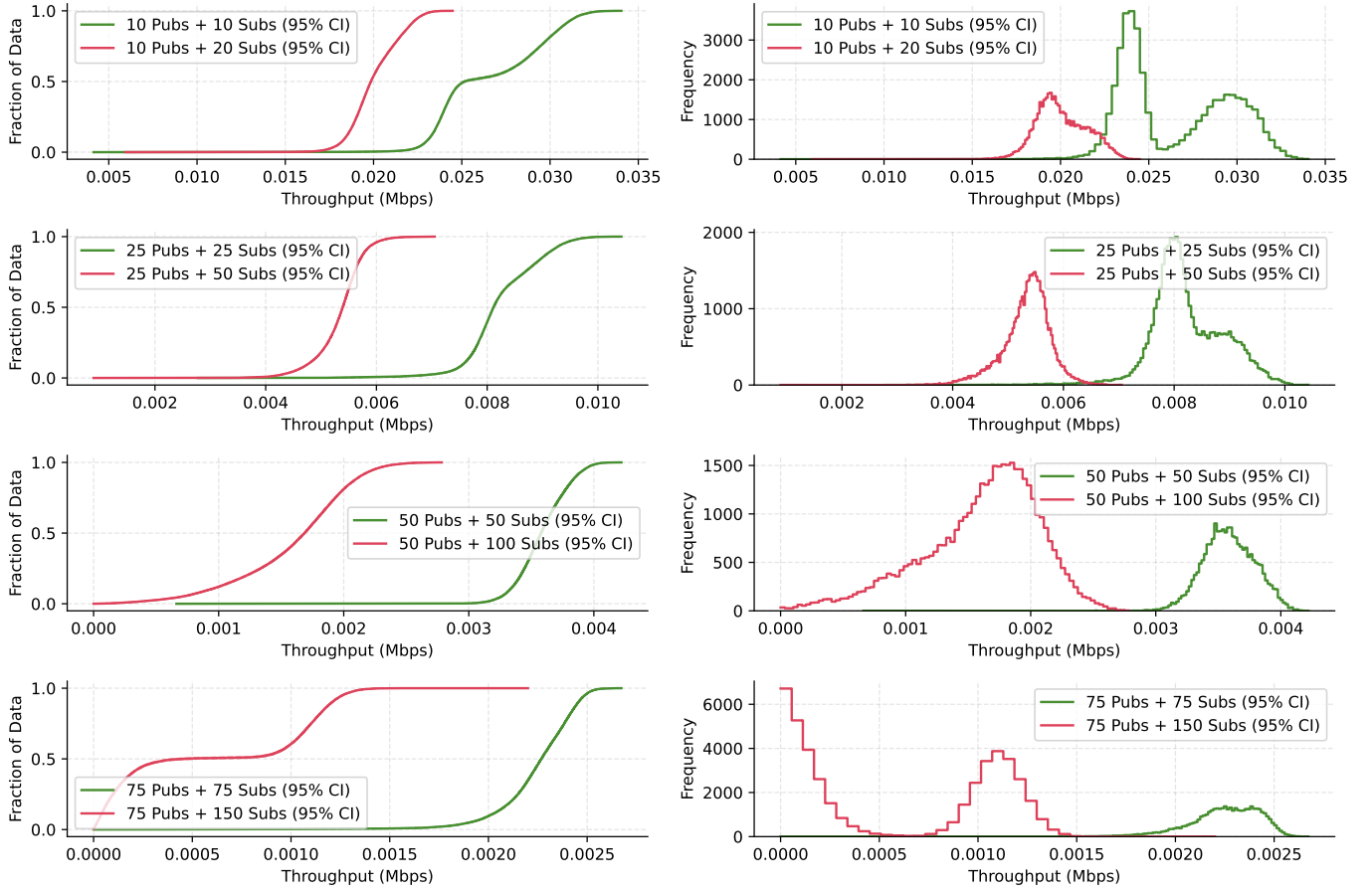


Figure 4.9: eCDF and PDF of throughput for malicious subscribers across different subscriber counts both individually and combined.

demonstrated with clear stochastic ordering seen in all eCDFs. The eCDFs indicate that systems with 25 and 50 subscribers exhibit comparable performance. A clear stochastic ordering is evident when comparing the 10-subscriber configuration against both the 25- and 50-subscriber configurations, with the latter 2 generally outperforming the former. Both the 25- and 50-subscriber configurations peaked at approximately the same throughput value, around 240 Mbps. However, the 50-subscriber configuration displayed a higher peak in its PDF, suggesting a greater concentration of instances operating at this high throughput value. These findings strongly suggest that the system’s throughput is capped at approximately 240 Mbps. Both the 25- and 50-subscriber configurations appeared to be reaching this performance limit. The results imply that the system likely encounters a performance ceiling at around 25 subscribers. This is particularly evident in the combined PDF plot, where the 50-subscriber configuration showed a significantly higher concentration of values around 220 Mbps compared to the 25-subscriber configuration, indicating a state of network saturation. This saturation point highlights a critical limitation in scaling the system’s throughput beyond a certain number of malicious subscribers.

4.2.5 Confirmation of Variation

The VMs were operating on a vLAN comprising 38 machines. It was hypothesised that other machines on the vLAN were interfering with the measurements and to confirm this, additional experiments were conducted. Details regarding the setup of this experimental campaign are provided in Subsection 3.4.3. Variations observed between runs suggested that external traffic was influencing the measurements.

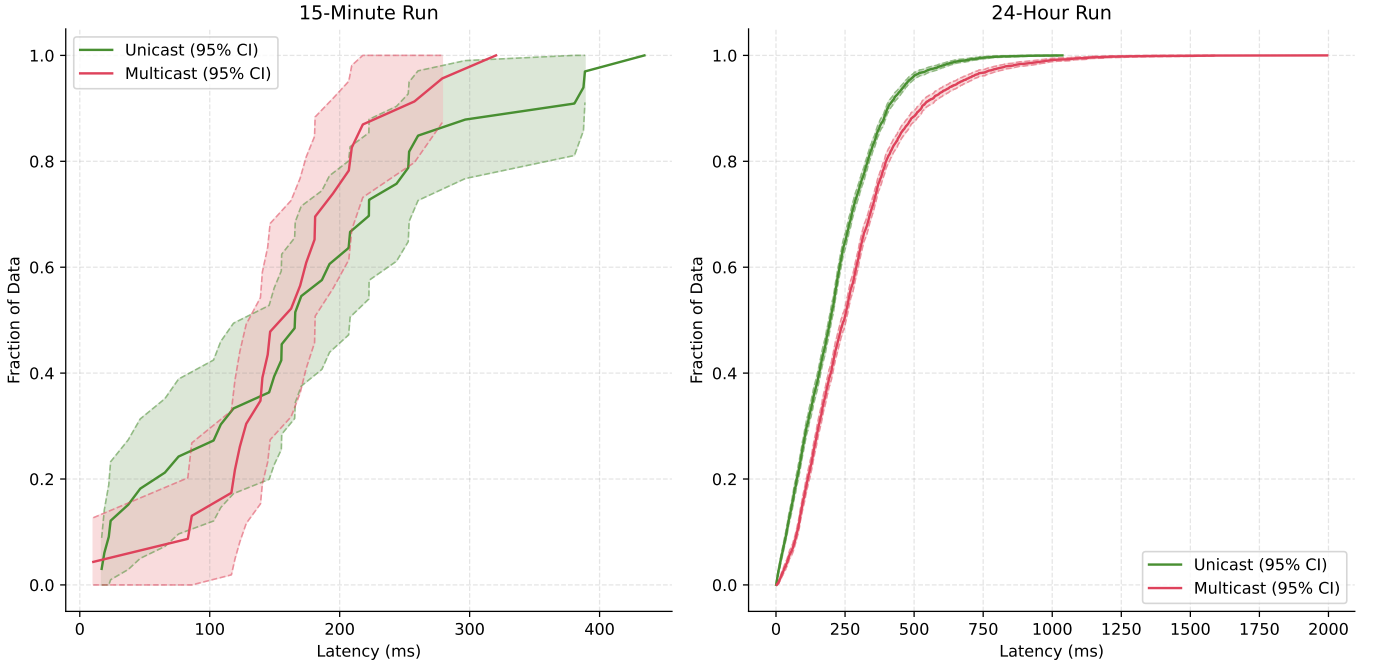


Figure 4.10: eCDFs with 95% confidence intervals for UC (green) and MC (red) latency (ms) of three 15-minute runs (left) versus three 24-hours runs (right).

Figure 4.10 shows the latency measurements for both the 15-minute and 6-hour configurations and Figure 4.11 compares the eCDF of latency and throughput for the variation confirmation campaign against the base case. These figures demonstrate variation between runs, indicating that external traffic on the vLAN is impacting the measurements. This variation is minimal for latency measurements and the overall pattern remains consistent. However, the variation is more pronounced in throughput measurements and as such does not provide a clear insight into the performance of DDS.

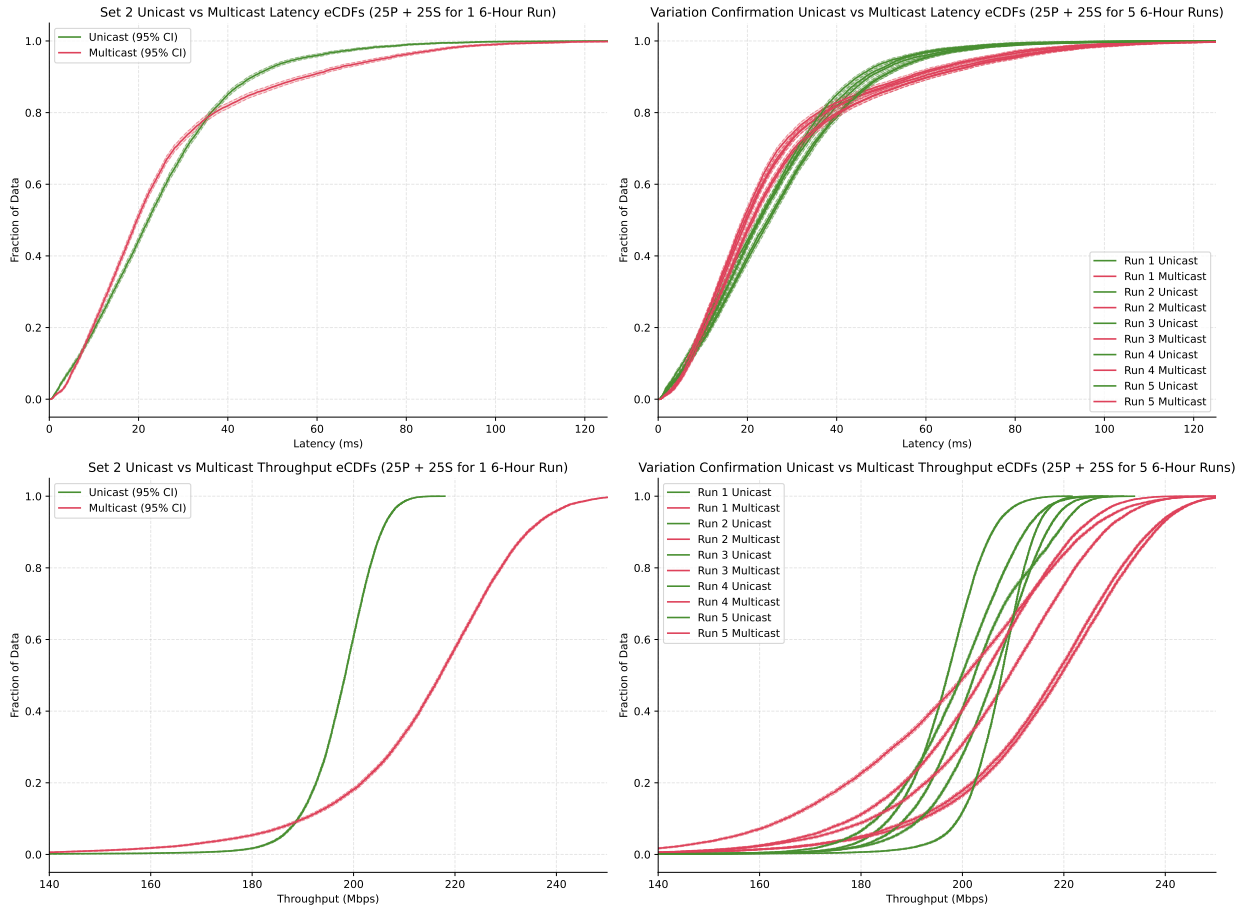


Figure 4.11: Comparing variation confirmation eCDF against base case for both latency and throughput. Top: latency. Bottom: throughput. Left: base case. Right: variation confirmation.

4.2.6 Single versus Multiple Machine

The results of the Single versus Multiple machine campaign detailed in Subsection 3.4.3 are presented in Figure 4.12. Negligible variation is observed in the multi-machine experimental setup while considerable variation is noted in the single-machine configuration which is counter-intuitive. Greater variation would be anticipated across multiple machines due to network involvement, yet the opposite is observed. The results suggest that messages may be exiting the single machine and returning to it, potentially causing the observed variability. These results in combination with those from the variation confirmation campaign in Subsection 4.2.5 indicate that external network traffic is impacting the measurements and evaluating DDS performance in virtualised environments is challenging due to uncontrollable external factors. A “cleaner” view of DDS performance is more attainable in physical environments without external interference.

Single Plot: Single vs Multi Machine Latency

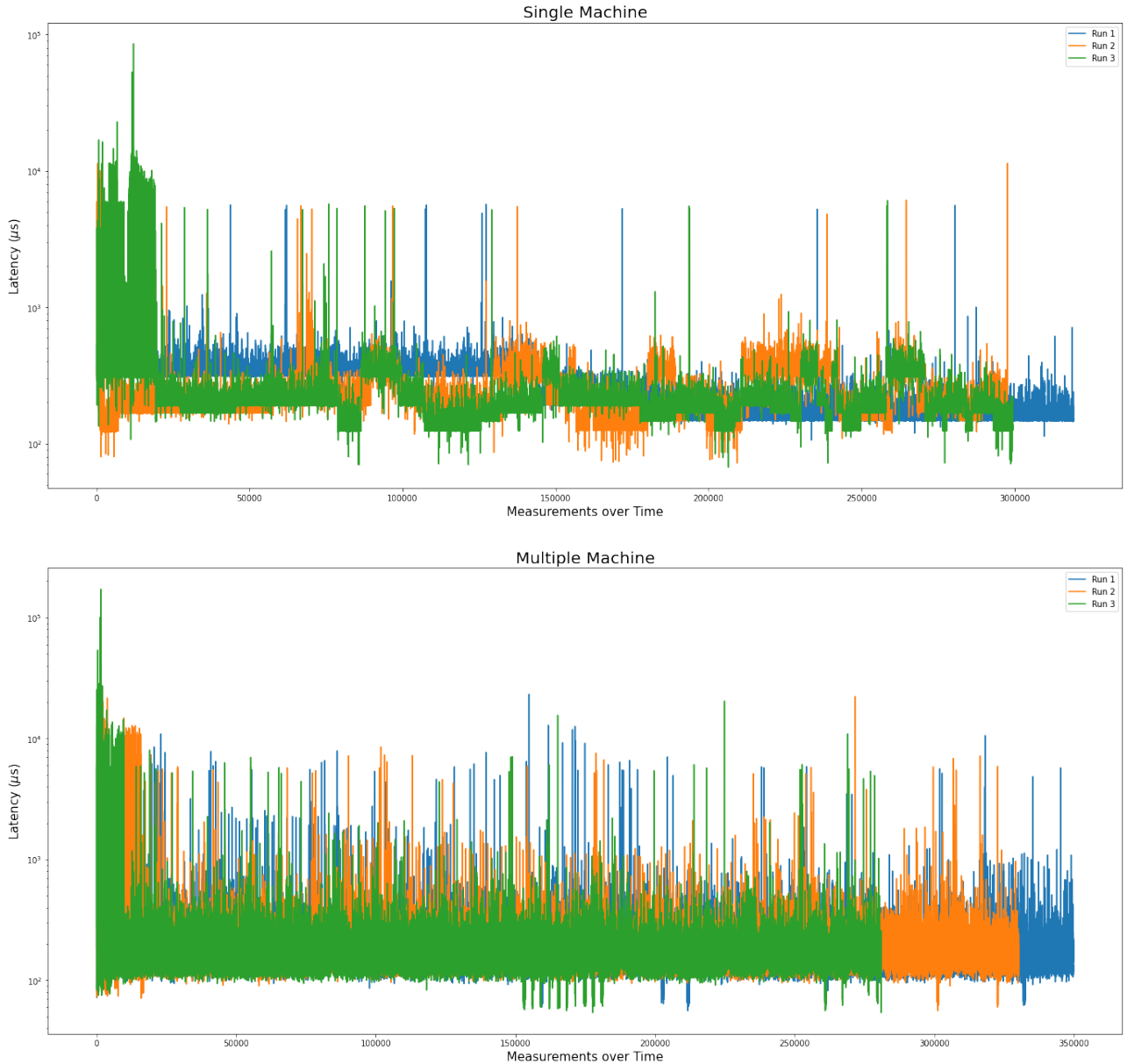


Figure 4.12: Latency over time for single (top) versus multiple machines (bottom) over 3 runs combined.

4.3 Analysis on the Impact of Multicast

4.3.1 Introduction

This section presents a detailed analysis of the impact of MC on the performance of the DDS. MC is a crucial feature for efficient communication to multiple subscribers, and this analysis aims to quantify its benefits and drawbacks under various conditions. UC communication defines that a host will only send data to another single host at any time while MC establishes the use of 1-to-n communication whereby a single host can send to multiple hosts in a single communication.

Despite not being defined within the DDS specification, the use of UC or MC still plays a vital role in how the communication of messages within DDS takes place. DDS users who have limited knowledge of the details of MC implementation and may assume that MC transport should, theoretically, outperform that of UC for greater numbers of participants since the data is being sent in one communication rather than multiple. This, however, does not necessarily hold in practice - a phenomenon that is explored in this section and is built on top of our work published in (Peeroo et al. 2022).

This section is structured as follows.

- Section 4.3.2 presents an initial analysis of the impact of MC on DDS performance using data collected on VMs (which was published in (Peeroo et al. 2022)).
- Section 4.3.3 builds on top of this by analysing the impact in a physical testbed (3Pi) within a segregated network.
- Section 4.3.4 and Section 4.3.5 further extends the analysis by examining data from a larger physical testbed (5Pi) with a more complex network topology.
- Section 4.3.6 provides a comparative analysis of the latency results across all testbeds used in this study and Section 4.3.7 does the same for throughput results.
- Section 4.3.8 analyses the impact across all dataset collected, broken down into quantiles of interest and compares eCDFs of both UC and MC communication using the PAT tool (see Subsection 3.3.4).

4.3.2 Initial Analysis

An initial analysis of the data is performed to get a high-level understanding of the impact of MC. This was done by analysing the data collected on the VM testbed and details of the experimental campaign can be found in Subsection 3.4.3. The counter-intuitive results produced were validated by Javier Morales, the head of the performance team at RTI and lead to the publication of our paper (Peeroo et al. 2022). This subsection presents the results of this initial analysis broken down into latency and throughput results.

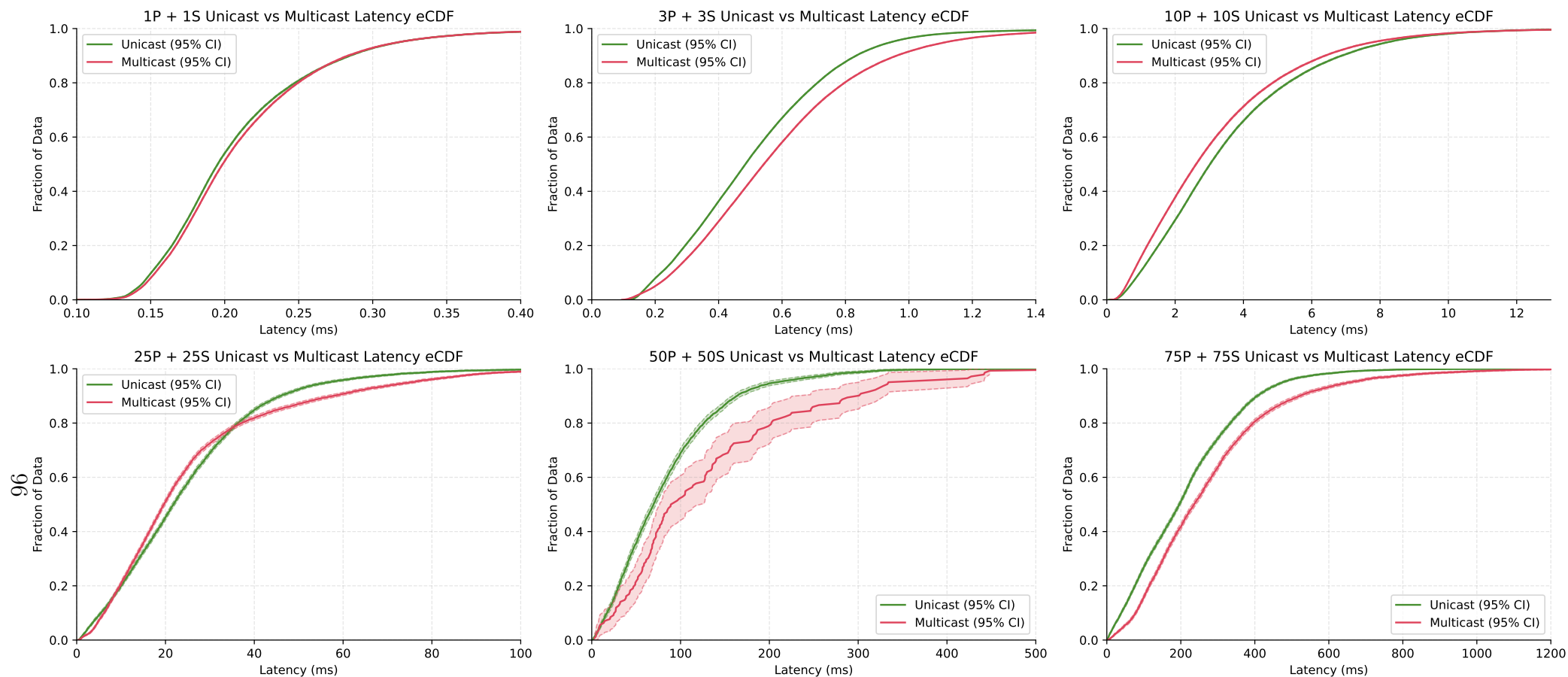


Figure 4.13: Latency eCDFs for varying participant numbers.

Latency Results

In experiments with 1 publisher and 1 subscriber, UC had very similar latency to MC with both communication types having a mean of 0.22 ms indicating that MC incurs a small overhead even in this simple scenario. In the experiments with 3 publishers and 3 subscribers, UC had a mean of 0.56 ms whilst MC had a mean of 0.62 ms - an 11% increase for MC over UC. For the experiments with 10 publishers and 10 subscribers, the ranking was reversed: the mean UC latency was 3.6 ms whilst the mean MC latency was 3.2 ms - an 11% decrease for MC over UC which shows the effectiveness of MC in this scenario. However, this effectiveness was not demonstrated in experiments with 25 publishers and 25 subscribers where similar performance was seen. Experiments with 50 publishers and 50 subscribers also showed stochastic dominance where MC consistently produced higher latencies. UC had a latency mean of 8.3 ms whilst MC had a latency mean of 12.5 ms - a 51% increase. This is not anticipated because MC is expected to perform better with larger numbers of participants. In the final experiment with 75 publishers and 75 subscribers, the distribution function shows consistently lower latency values for UC (with a mean of 312 ms) compared to MC (having a mean of 427 ms - a 37% increase). Again, this is counter-intuitive as MC is expected to perform better with larger numbers of participants.

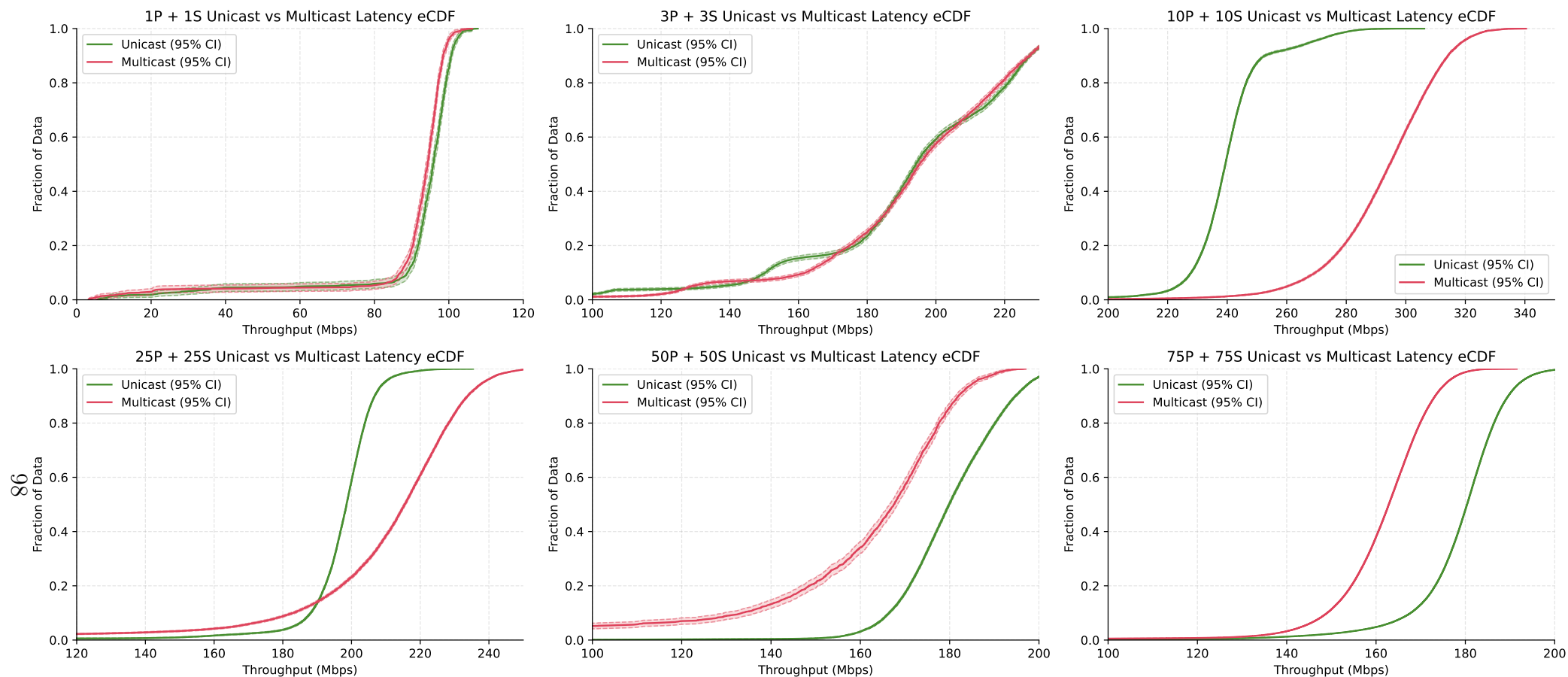


Figure 4.14: Throughput eCDFs for varying participant numbers.

Throughput Results

Throughput measurements are presented as eCDFs in Figure 4.14 for 1, 3, 10, 25, 50 and 75 publishers/subscribers. The experiments involving 1 publisher and 1 subscriber show that roughly the first 10% of the ordered throughput measurements were almost identical. For throughput measurements above 85 Mbps, making up a majority of all measurements, UC produced higher throughput values than MC. In the experiments with 3 publishers and 3 subscribers, neither communication setting was dominant, while MC outperforms UC for 10 publishers and 10 subscribers with a 23% increase in the mean throughput (237 Mbps for UC compared to 292 Mbps for MC). Similar performance was also seen for 25 publishers and 25 subscribers for the first roughly 10% of the ordered throughput measurements and MC outperforms UC for the rest of the measurements above 190 MBps. The performance gain for MC over UC disappears in the experiments with 50 and 75 publishers/subscribers where UC outperforms MC by 6% and 9% according to the mean values, respectively.

Analysis

Publisher Count	Subscriber Count	Latency	Throughput
1	1	UC	UC
3	3	UC	mixed
10	10	MC	MC
25	25	mixed	MC
50	50	UC	UC
75	75	UC	UC
Total	UC	4	3
	MC	1	2
	mixed	1	1
	Total	6	6

Table 4.3: Most performant communication type per participant configuration for both latency and throughput on the VM testbed.

When a large number of participants uses MC communication, one can expect that all of the recipients' data are sent using one communication rather than multiple (as in the case of UC) which should theoretically result in better performance (lower latency and higher throughput). The results, however, demonstrate the opposite in several cases. A summary showing the most performant communication is shown in Table 4.3.

The inferior performance of MC in high-participant scenarios could be attributed to several factors. Firstly, MC introduces a higher protocol overhead compared to UC. This includes the

management of MC groups and the processing of Internet Group Management Protocol (IGMP) messages by network switches. While this overhead is negligible in small-scale deployments, it can become significant as the number of participants grows, potentially outweighing the benefits of 1-to-n communication.

Secondly, the performance of MC is highly dependent on the capabilities of the underlying network hardware, particularly the switches. Inexpensive or unmanaged switches may not handle MC traffic efficiently, leading to packet flooding and network congestion. Even with managed switches, the efficiency of MC is contingent on the proper implementation and configuration of IGMP snooping, which allows switches to intelligently forward MC traffic only to the ports that need it. Without effective IGMP snooping, MC packets can be broadcast to all ports, negating the benefits of MC and potentially leading to worse performance than UC.

Thirdly, the use of UDP for MC transport in DDS means that there is no built-in mechanism for reliable delivery at the transport layer. While DDS provides its own reliability protocol, the underlying best-effort nature of MC can lead to higher packet loss rates in congested networks, which in turn can trigger retransmissions at the DDS layer, increasing latency and reducing throughput.

After investigating the cause of this, we conjecture that performance gained when using MC communication with DDS largely depends on the network setup. Specifically, which MC routing protocol is being used. The main point from this work is that one must have a thorough understanding of how MC communication is implemented on one's (experimental) setup before using it for DDS deployment. One should not assume that using MC via a DDS implementation with large numbers of participants will necessarily result in better performance. The results may not be representative of a real-world scenario where the network topology and MC routing protocols may differ significantly and a critical evaluation of the network infrastructure and the specific DDS implementation is essential for making an informed decision about when to use MC. This analysis is extended in subsequent sections by examining data from physical testbeds with different network topologies and configurations to uncover the conditions under which MC can be expected to outperform UC in DDS deployments.

4.3.3 Analysis on the 3 Raspberry Pi Multicast Dataset

New experiments (detailed in Subsection 3.4.4) on the 3Pi testbed were executed to validate and further investigate the initial analysis. These added experiments used 100 B or 512 B, reliable communication, a durability level of 0, and explored new combinations of participant

counts including cases with a single publisher/subscriber. The results are therefore grouped into 3 sections (where $n > 1$):

1. Analysis of n Publishers and n Subscribers
2. Analysis of 1 Publisher and n Subscribers
3. Analysis of n Publishers and 1 Subscriber

The Analysis of n Publishers and n Subscribers section matches the publisher/subscriber counts used in the initial analysis on the VM testbed. The following section, Analysis of 1 Publisher and n Subscribers, explores the impact of MC in its theoretically most advantageous scenario where a single publisher sends data to multiple subscribers. The final section, Analysis of n Publishers and 1 Subscriber, investigates the reverse scenario where multiple publishers send data to a single subscriber to investigate if MC has any benefits in this case. Finally, a summary of all results related to the 3Pi testbed is presented in Summary of 3Pi Multicast Analysis to provide a wholistic view.

Analysis of n Publishers and n Subscribers

Latency Figure 4.15 displays eCDFs that compare MC and UC latency for configurations with 1, 5, 10, 15, and 25 publishers and subscribers. MC achieves lower latencies than UC in experiments involving 1, 5, and 10 publishers and subscribers, though in the case with 10 publishers/subscribers, there is some overlap at the lower percentiles. Similar performance is seen for 15 publishers/subscribers, and UC outperforms MC in the 25-participant experiment. As the number of participants increases, the gap between the two communication types narrows before reversing in favour of UC at 25 publishers and subscribers which is counter-intuitive as one would expect MC to perform better with larger numbers of participants. This supports the findings from the initial analysis on the VM testbed in Subsection 4.3.2.

Throughput The analysis of throughput, illustrated by the eCDFs in Figure 4.16, reveals more cases in favour of MC. The case with 1 publisher/subscriber is the only scenario where both communication types show overlapping performance. For all other cases (5, 10, 15, and 25 publishers/subscribers), MC consistently outperforms UC, demonstrating stochastic ordering with a clear separation favouring MC. An interesting observation is the larger number of 0 Mbps values specifically for UC which increases with the number of participants. This is seen by the “rising” of the start of the UC eCDF shown in green at the far left of the graph as the number of participants increases from 10 to 25 publishers/subscribers. Not only is MC producing higher throughput, but it also appears to be more resilient to network delays that

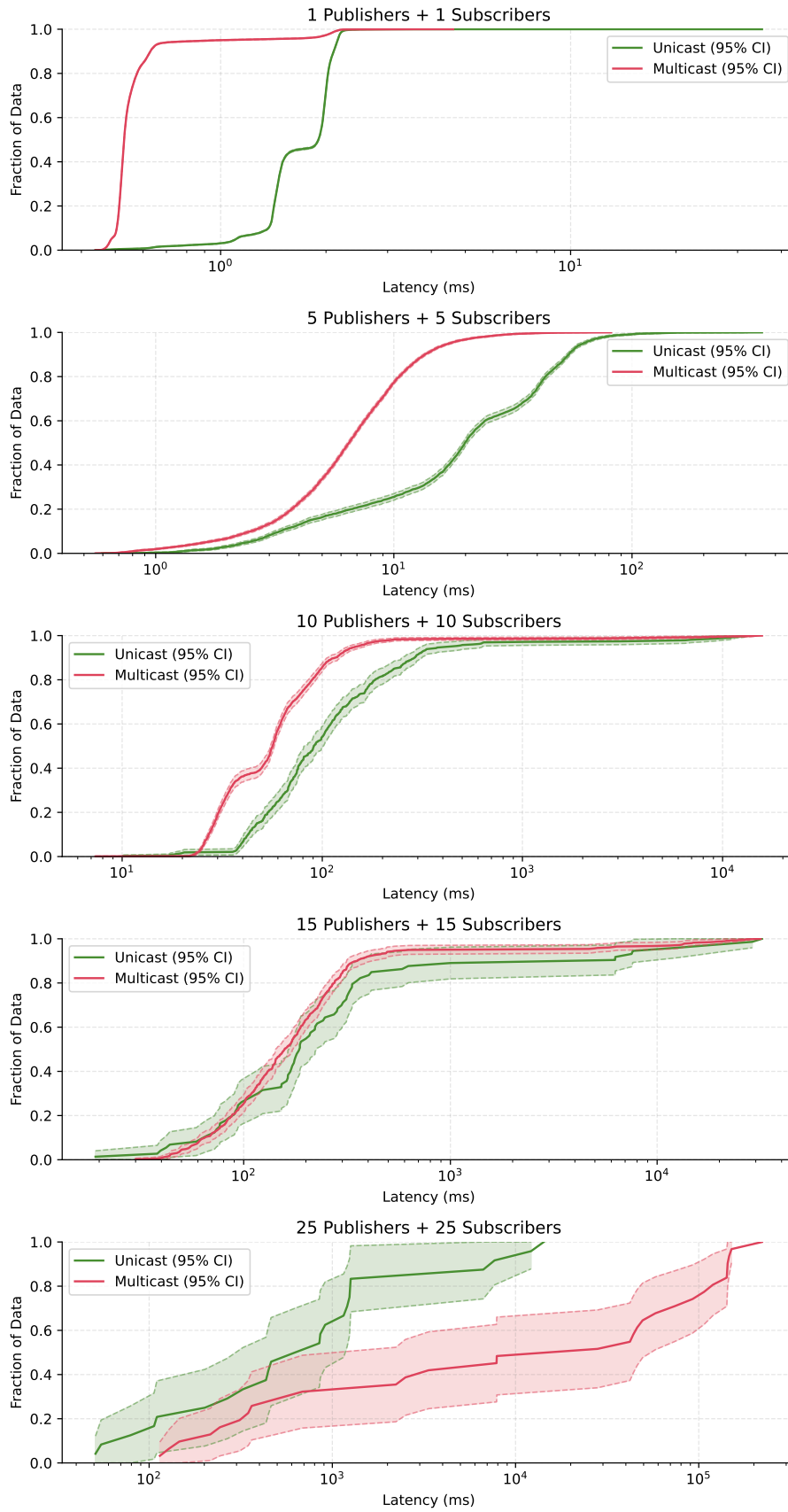


Figure 4.15: eCDFs of MC (red) and UC (green) latency for 1, 5, 10, 15, and 25 publishers and subscribers for the 3Pi MC dataset. The x-axis is in log scale.

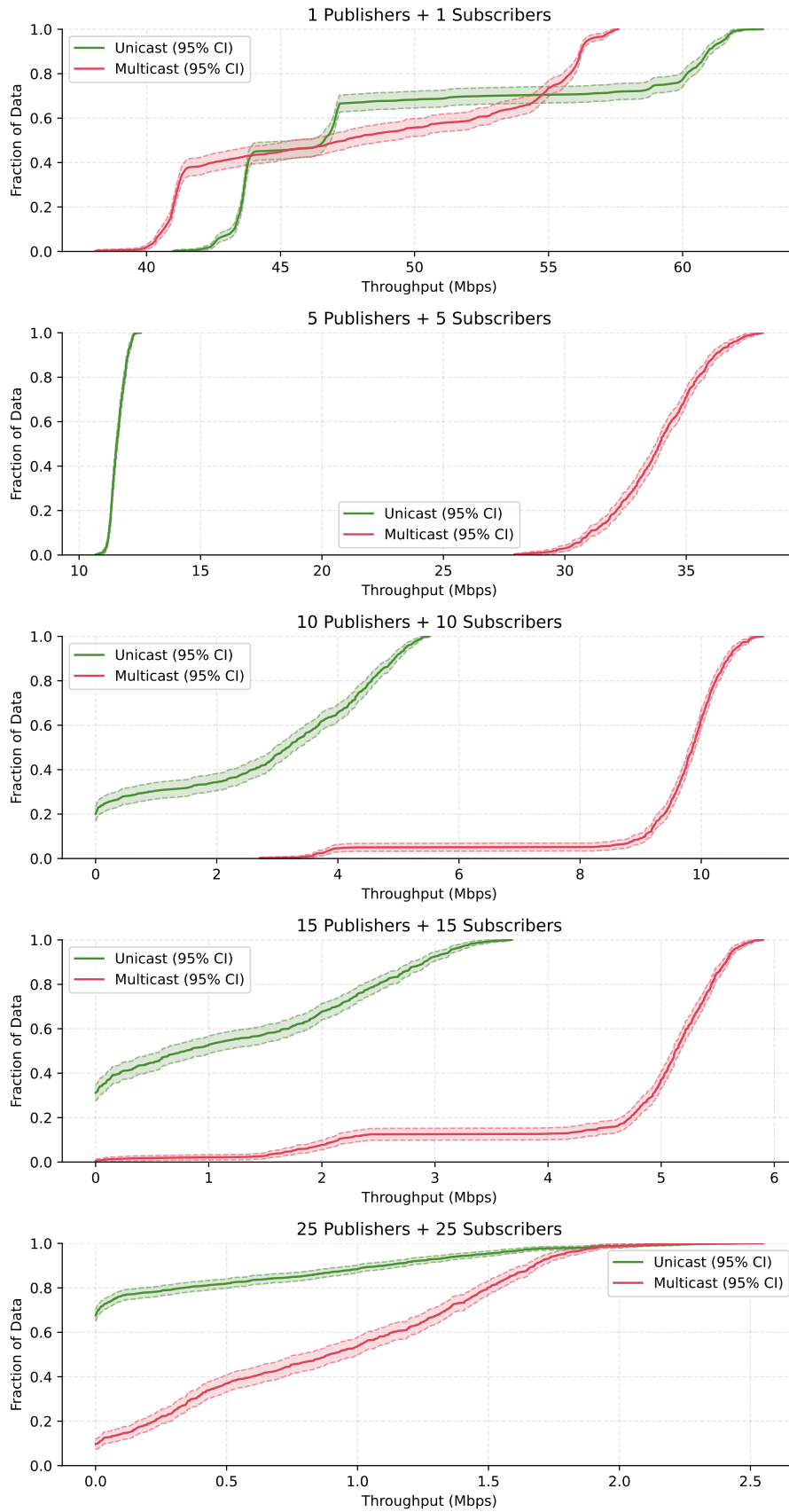


Figure 4.16: eCDFs of MC (red) and UC (green) throughput for 1, 5, 10, 15, and 25 publishers and subscribers for the 3Pi MC dataset.

result in 0 Mbps throughput values. The results here align with the expectation that MC should excel in scenarios with multiple subscribers due to its design for efficient data dissemination and do not conform to the counter-intuitive findings from the initial analysis on the VM testbed in Subsection 4.3.2.

Analysis of 1 Publisher and n Subscribers

Latency Figure 4.17 demonstrates a clear and consistent advantage for MC communication across all tested subscriber counts (1, 5, 10, 15, and 25). The gap between the eCDFs are similar for 5, 10, and 25 subscribers, and are much larger for 1 and 15 subscribers which might point to some underlying network behaviour specific to those configurations and warrants further investigation. Nonetheless, these results confirm the expectation that MC should excel in 1-to- n communication scenarios.

Throughput Figure 4.18 demonstrates that MC outperforms UC in terms of throughput for 5, 10, and 15 subscribers with clear stochastic ordering and this ordering switches in the case of 25 subscribers where UC outperforms MC with some intersection at higher percentiles. This is unexpected as one would expect MC to perform better with larger numbers of subscribers and hints at a potential bottleneck.

Analysis of n Publishers and 1 Subscriber

Latency Figure 4.19 shows that MC latencies consistently outperform UC with stochastic ordering though the distinction is not as pronounced in the case with 5 publishers. Nonetheless, these results demonstrate a clear advantage for MC in terms of latency in n -to-1 communication scenarios which is unexpected. What is particularly intriguing is that the gap in performance increases with the number of publishers indicating that MC becomes increasingly advantageous as more publishers are added.

Throughput The throughput results for the n -to-1 configuration are presented in Figure 4.20 and show a very similar pattern for 5, 10, 15, and 25 publishers where both communication types exhibit similar performance throughout. However, the gap is more pronounced in favour of MC for 25 publishers indicating that MC becomes increasingly advantageous as more publishers are added, a trend also seen in the latency results.

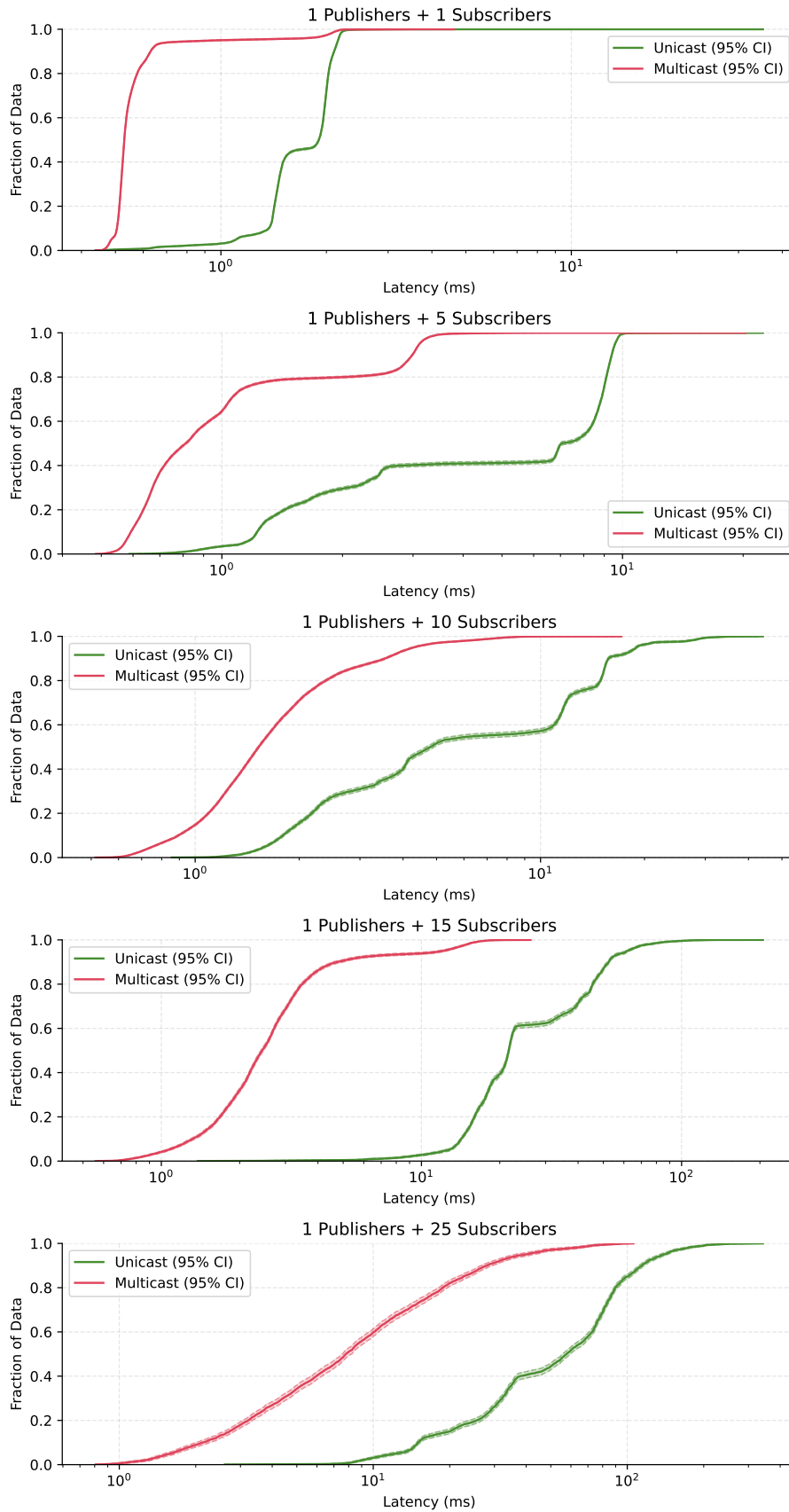


Figure 4.17: eCDFs of MC (red) and UC (green) latency for 1 publisher and 1, 5, 10, 15, and 25 subscribers for the 3Pi MC dataset. The x-axis is in log scale.

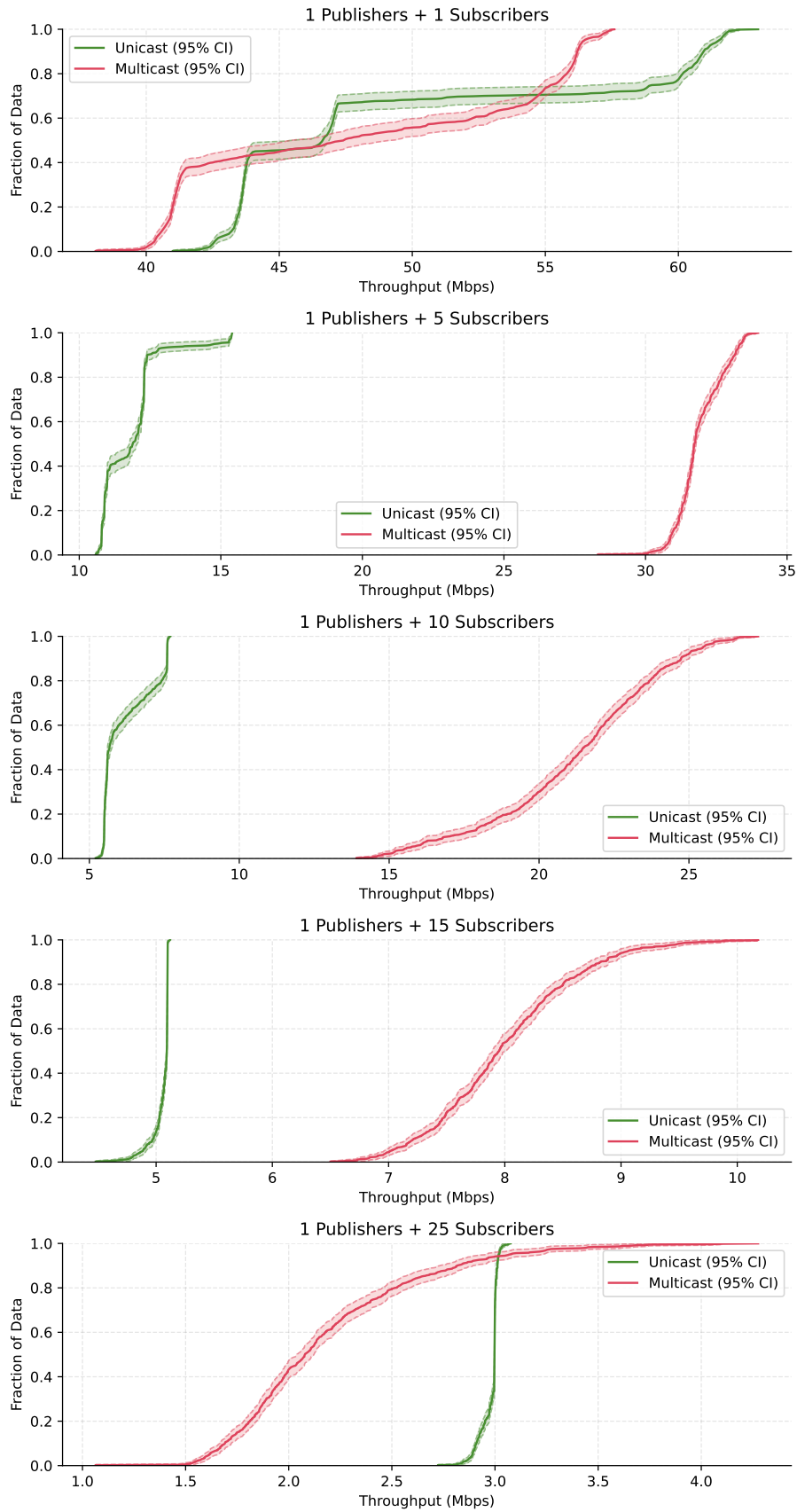


Figure 4.18: eCDFs of MC (red) and UC (green) throughput for 1 publisher and 1, 5, 10, 15, and 25 subscribers for the 3Pi MC dataset.

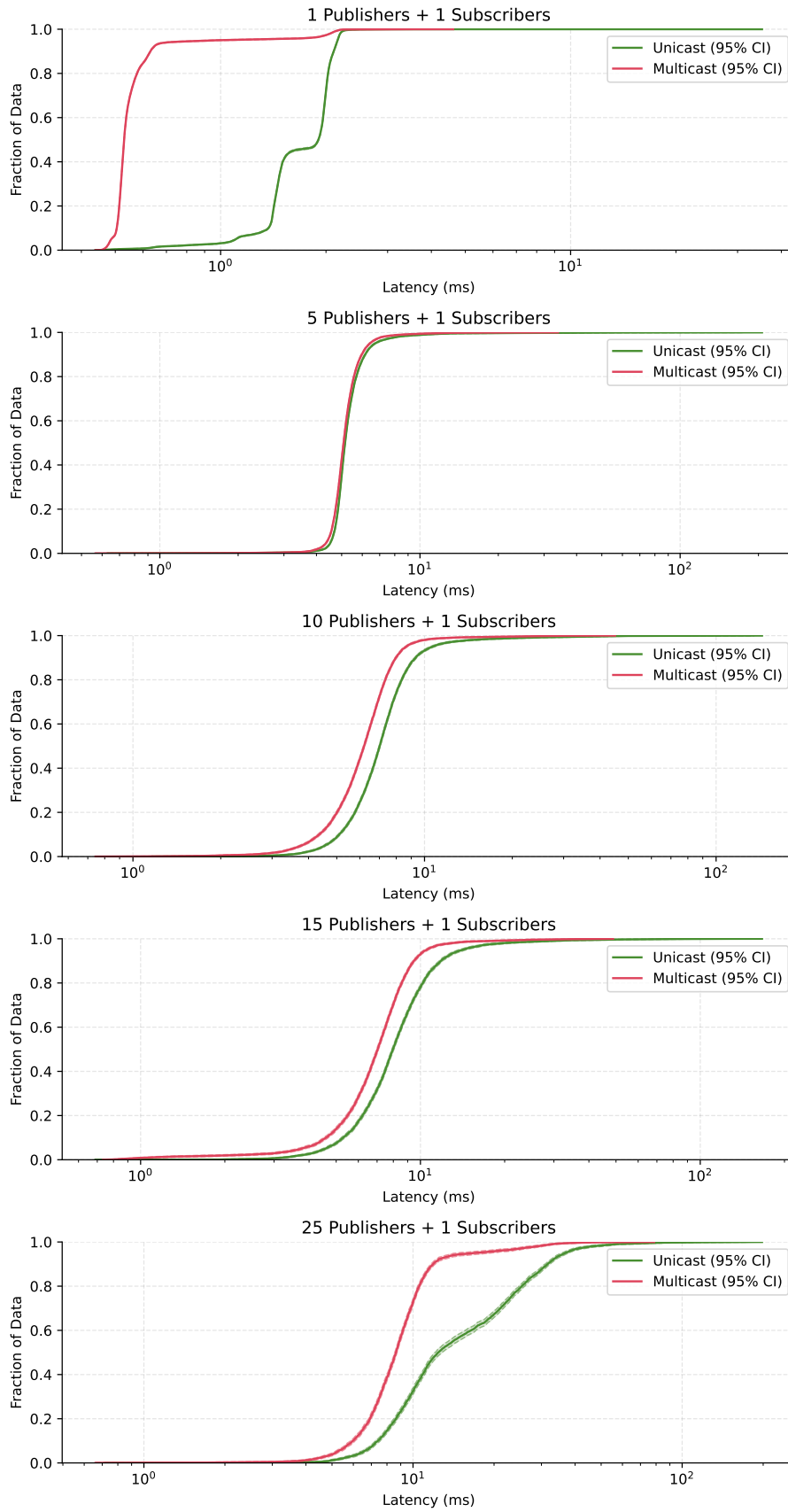


Figure 4.19: eCDFs of MC (red) and UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers for the 3Pi MC dataset. The x-axis is in log scale.

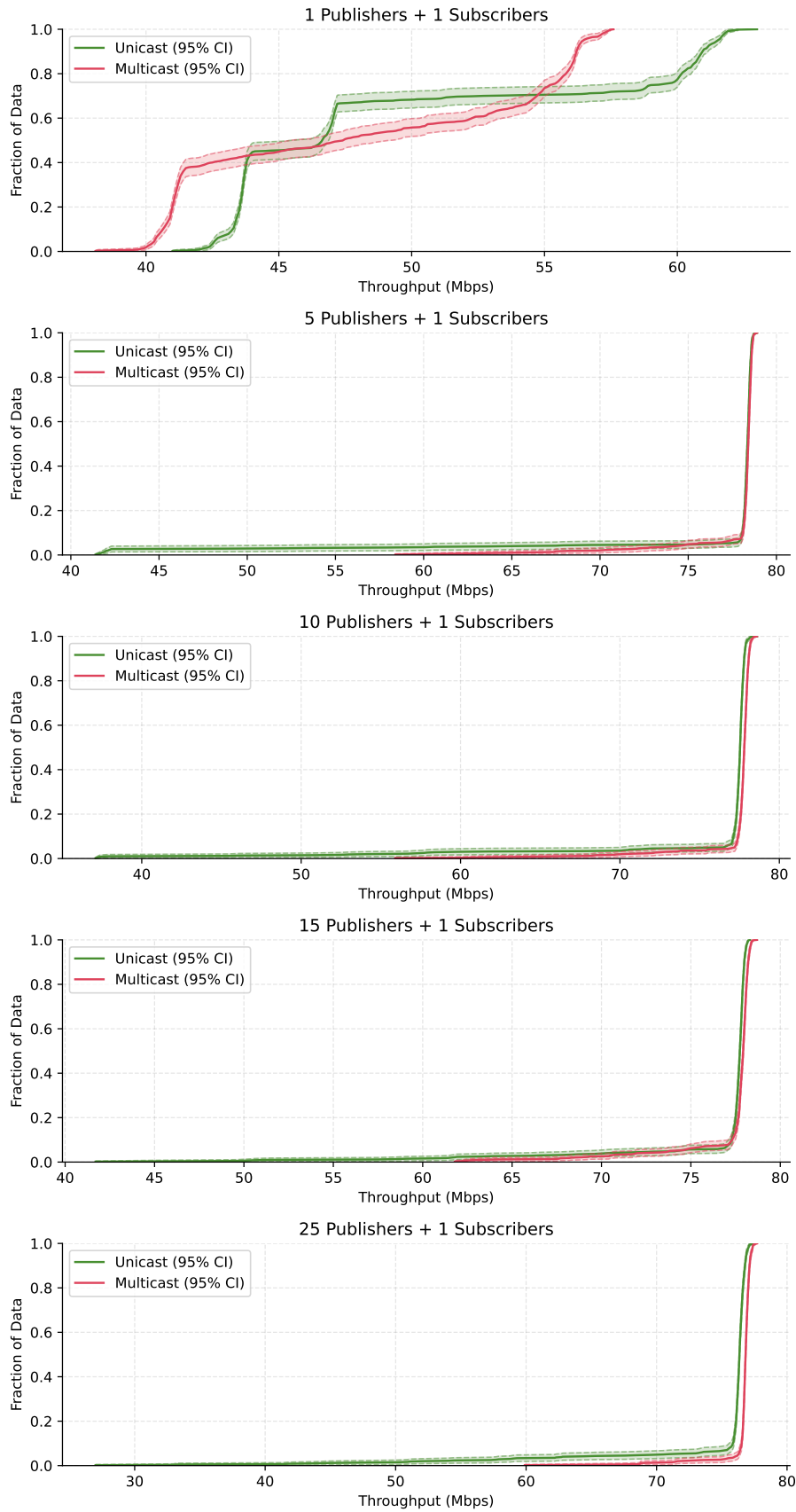


Figure 4.20: eCDFs of MC (red) and UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers for the 3Pi MC dataset.

Summary of 3Pi Multicast Analysis

Table 4.4 summarises the results of the analysis by showing the most performant communication type (either UC or MC) for each combination of publisher and subscriber count. MC outperforms UC in 10 cases for both latency and throughput while UC only outperforms MC in 1 case for throughput and none for latency. The large number of cases in favour of MC indicates that MC is more effective in the 3Pi testbed.

Publisher Count	Subscriber Count	Latency	Throughput
1	1	MC	none
1	5	MC	MC
1	10	MC	MC
1	15	MC	MC
1	25	MC	mostly UC
5	1	none	none
5	5	MC	MC
10	1	MC	MC
10	10	MC	MC
15	1	MC	mostly MC
15	15	none	MC
25	1	MC	MC
25	25	none	mostly MC
Total	UC	0	1
	MC	10	10
	none	3	2

Table 4.4: Most performant type of communication (UC or MC) in terms of latency and throughput for each combination of publisher and subscriber count according to visual stochastic ordering seen when comparing eCDFs. Mostly UC/MC indicates that there was some intersection seen in the eCDF. None indicates that there was no clear difference between the 2 communication types. Values in bold indicate cases with the same outcomes as seen in VMs.

Direct comparisons between these results and those collected on the VMs cannot be made because the testbeds are different (virtualised vs physical), the network setups differ (shared vs segregated) and the hardware specifications are not the same (VM vs RPi 3A+). Nonetheless, comparisons are made for 1, 10, and 25 publishers and subscribers since these configurations were used in both experimental campaigns to identify any similarities or differences in trends. Similarities are seen for 10 publishers and subscribers where MC outperforms UC for both latency and throughput. This is also the case for 25 publishers and subscribers where MC outperforms UC for throughput. These similarities may be indicative of a trend and we analyse 5Pi data in the next subsection to further investigate this.

4.3.4 Analysis on the 5 Raspberry Pi Multicast Dataset

Similar to the experiments on the 3Pi testbed that investigated the impact of MC, new experiments (detailed in Subsection 3.4.5) were executed on the 5Pi testbed to further validate and investigate the findings from both the initial analysis on VMs (Subsection 4.3.2) and the 3Pi testbed (Subsection 4.3.3). These new experiments used 512 B, reliable communication, a durability level of 0, and explored values of 1, 5, 10, 15, 20, and 25 publishers/subscribers.

Just like in the 3Pi analysis, we break down the results into 3 sections (where $n > 1$):

1. Analysis of n Publishers and n Subscribers
2. Analysis of 1 Publisher and n Subscribers
3. Analysis of n Publishers and 1 Subscriber

Analysis of n Publishers and n Subscribers

Latency The latency results for the n-to-n configuration on the 5Pi testbed are presented in Figure 4.21 and shows a clear pattern in favour of MC with intersection seen at higher percentiles in the case of 15 publishers and subscribers. This is especially interesting for the case with 1 publisher and subscriber where MC outperforms UC which is unexpected as one would expect both communication types to perform similarly in this simple scenario.

Throughput The throughput results for the n-publisher-to-n-subscriber configuration are shown in Figure 4.22 and do not show a clear pattern similar to the latency results. MC is seen to outperform UC in the cases with 5, 10, and 15 publishers and subscribers, though there is some intersection for the case with 5 publishers and subscribers. However, in the case with 1 publisher and subscriber, UC outperforms MC which is unexpected as one would expect both communication types to perform similarly in this simple scenario. A noticeable step-like pattern is seen in the case of 5 and 10 publishers and subscribers which may point to some clustering of throughput values possibly due to network behaviour.

Analysis of 1 Publishers and n Subscribers

Latency The latency results for the 1-to-n configuration on the 5Pi testbed are presented in Figure 4.23 and are unequivocal: MC consistently demonstrates lower latency than UC across all subscriber counts and this is consistent with the findings from the 3Pi testbed. The eCDFs show stochastic ordering in favour of MC, and the performance gap between the communication

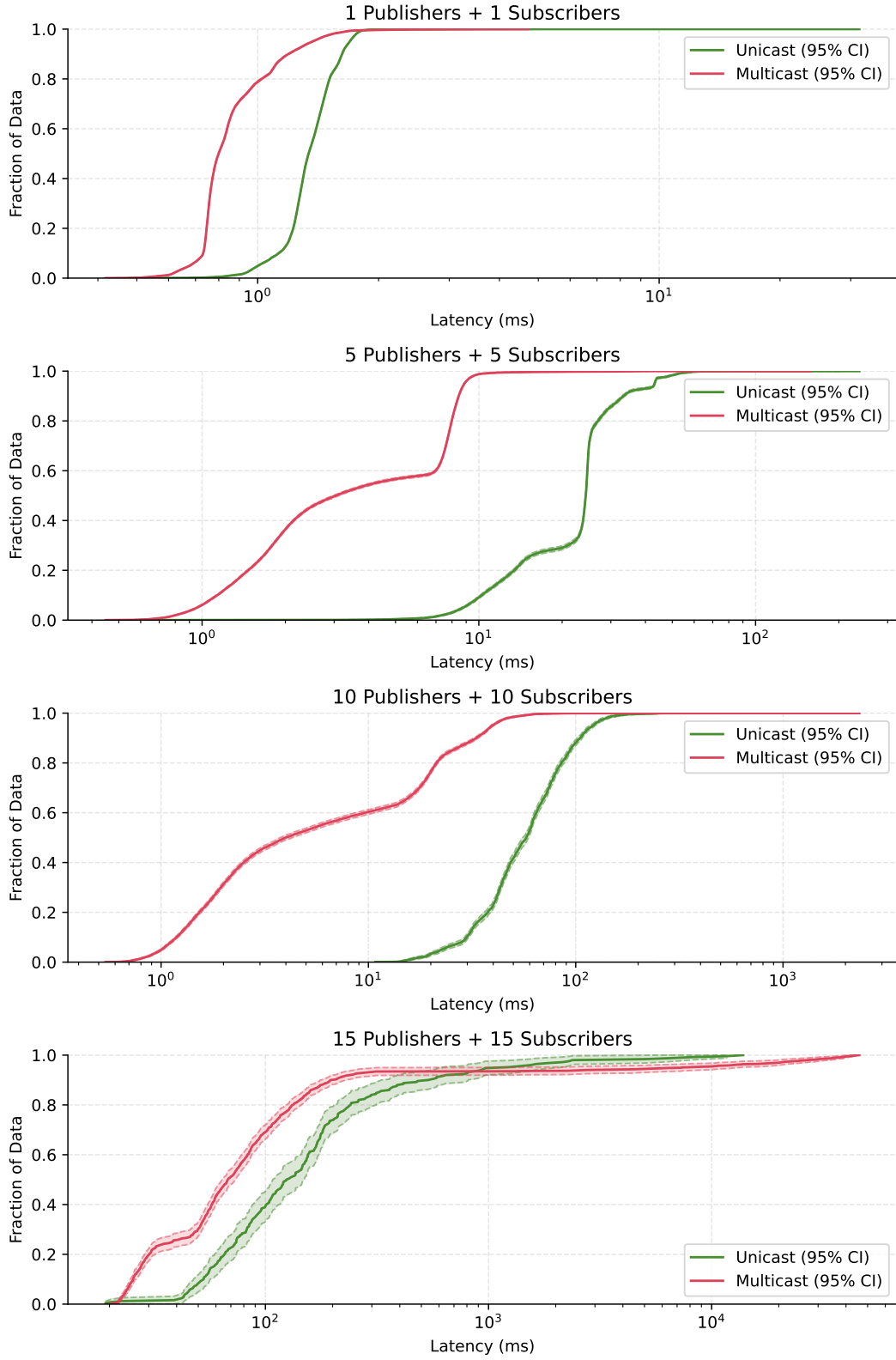


Figure 4.21: eCDFs of MC (red) and UC (green) latency for 1, 5, 10, and 15 publishers and subscribers for the 5Pi MC dataset. The x-axis is in log scale.

methods is even more pronounced on the 5Pi. These results strongly support the theoretical expectation that MC should excel in 1-to-n communication scenarios.

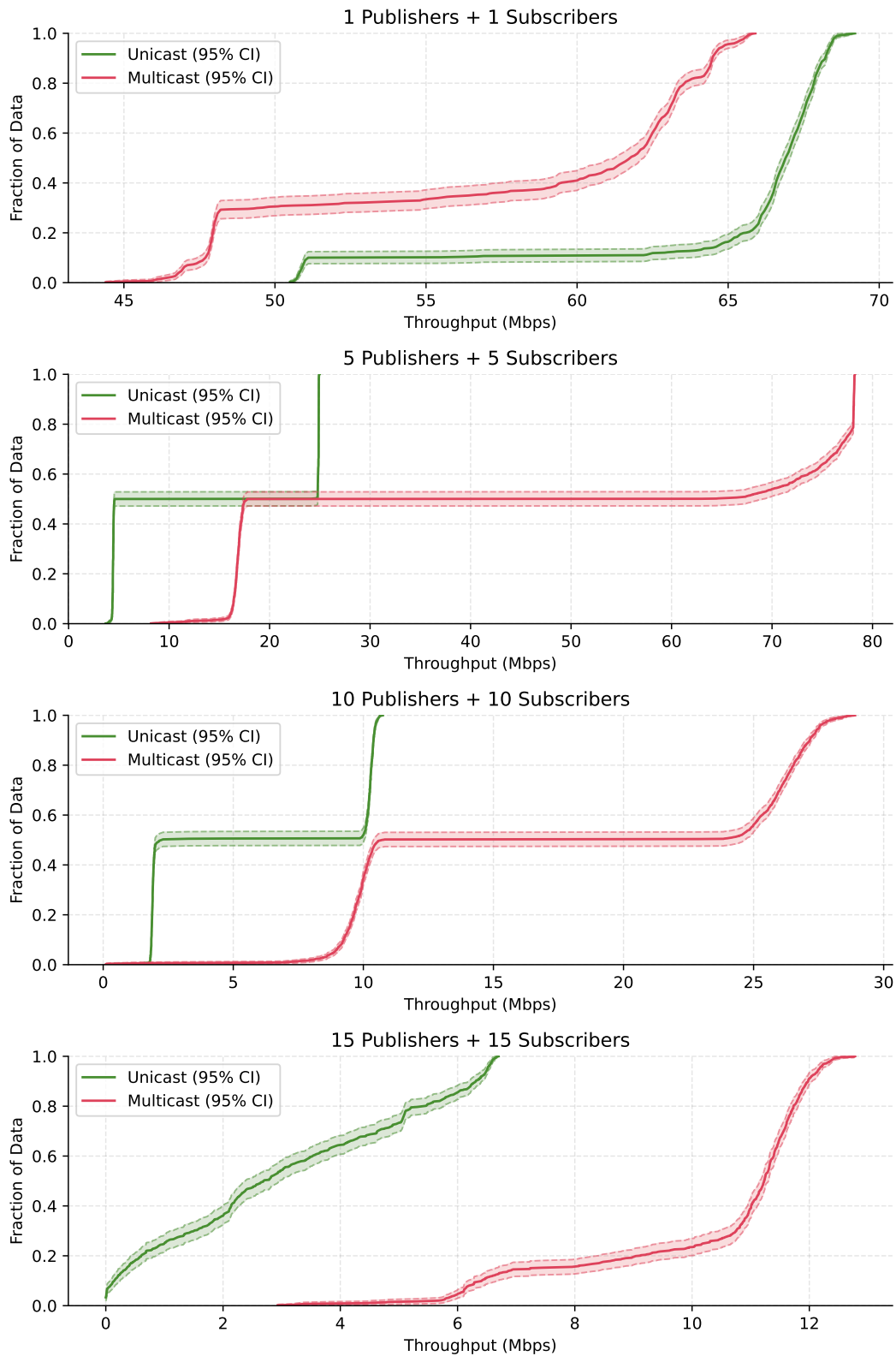


Figure 4.22: eCDFs of MC (red) and UC (green) throughput for 1, 5, 10, and 15 publishers and subscribers for the 5Pi MC dataset.

Throughput The throughput results, shown in Figure 4.24, also show a clear advantage for MC in all cases with more subscribers and 1 publisher though there is intersection in the case of 1 publisher and 5 subscribers. This indicates that for 1-to-n communication on the 5Pi testbed,

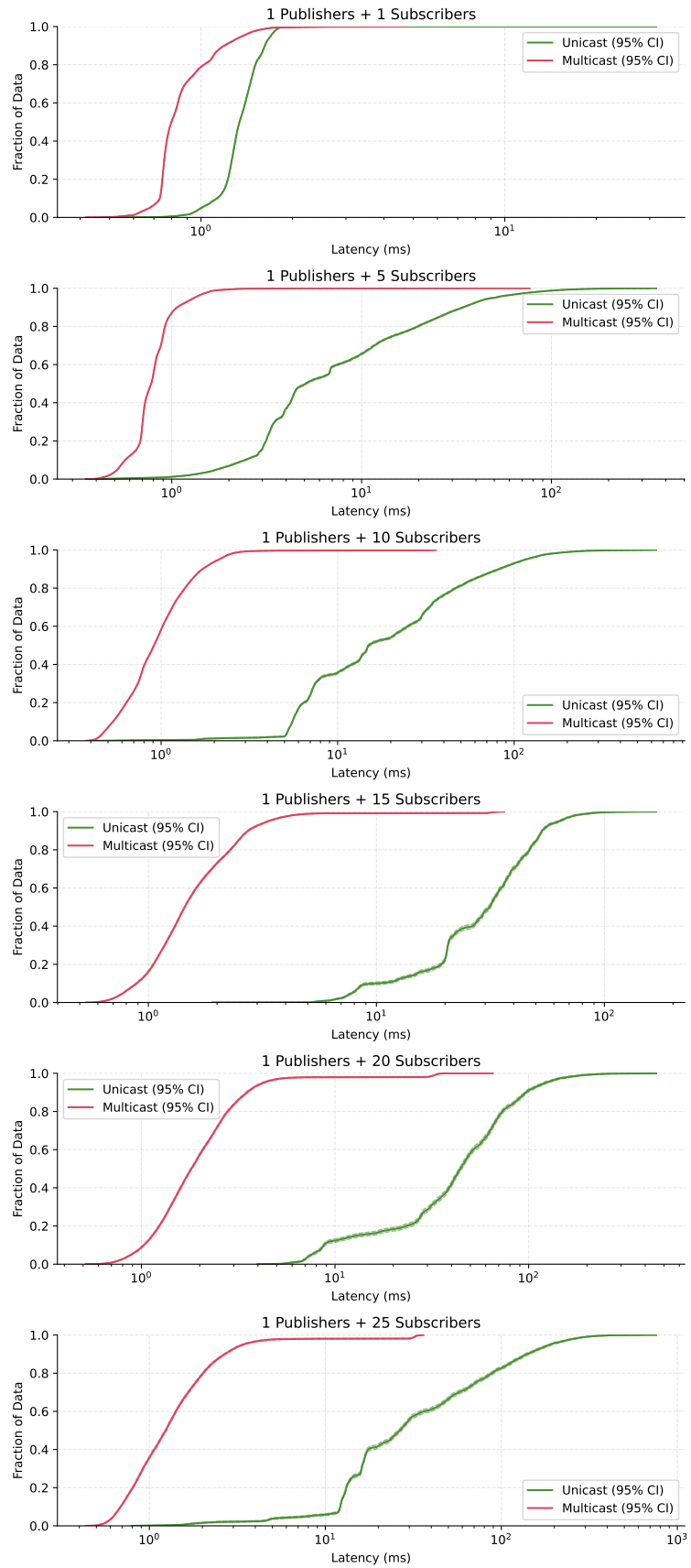


Figure 4.23: eCDFs of MC (red) and UC (green) latency for 1 publisher and 1, 5, 10, 15, 20, and 25 subscribers for the 5Pi MC dataset. The x-axis is in log scale.

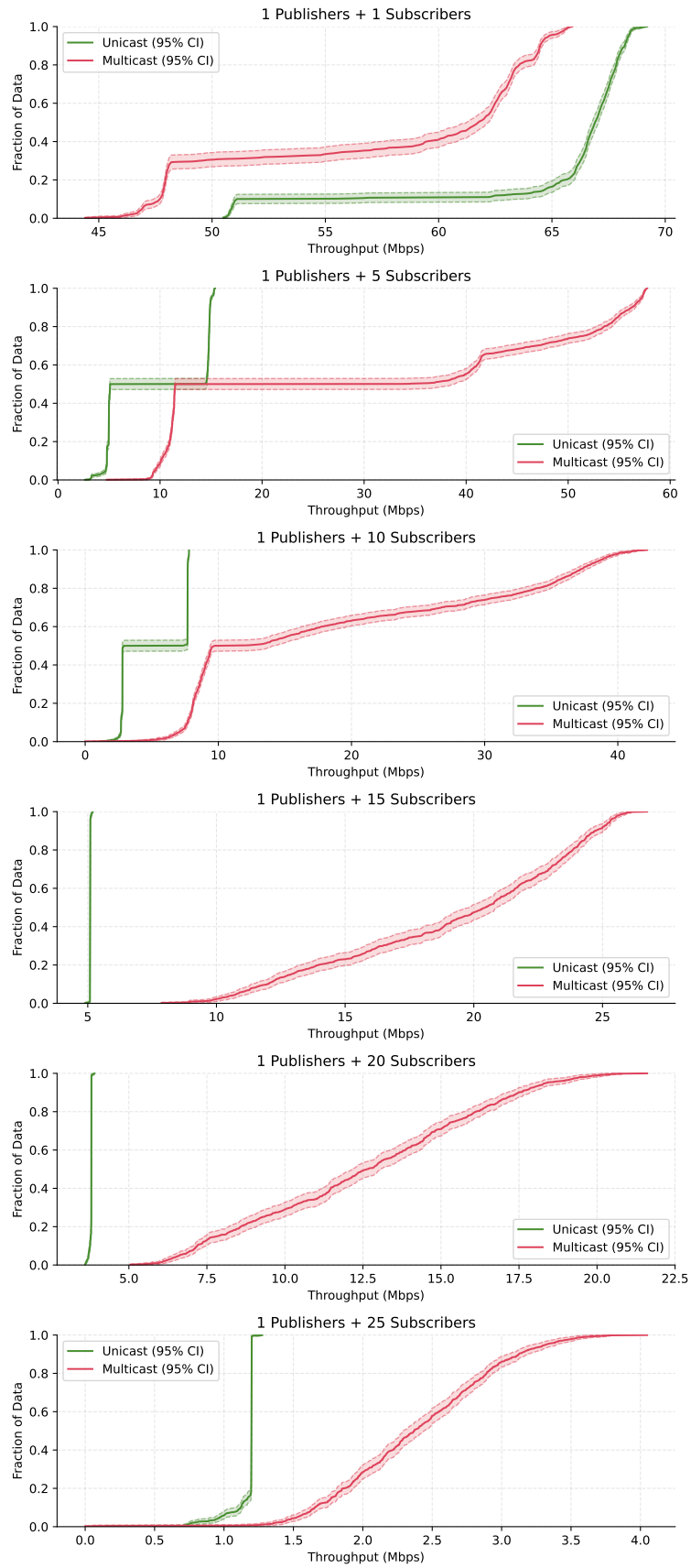


Figure 4.24: eCDFs of MC (red) and UC (green) throughput for 1 publisher and 1, 5, 10, 15, 20, and 25 subscribers for the 5Pi MC dataset.

MC is the superior choice for maximising throughput.

Analysis of n Publishers and 1 Subscriber

Latency The latency results for the n -to-1 configuration on the 5Pi testbed are presented in Figure 4.25 and shows the same trend across all publisher counts; both communication types produce extremely similar performance. Step-like patterns emerge in the case with 15 and 20 publishers which may point to some clustering of latency values possibly due to network behaviour. This hints at two possible states in the network that cause latency to cluster around two distinct values.

Throughput The throughput results for the n -to-1 configuration are shown in Figure 4.26 and similar to the latency results, show very similar performance between MC and UC. In the cases with 15 and 20 publishers, UC outperforms MC with intersection at lower and higher percentiles. This performance gain for UC over MC indicates that the subscriber is receiving more data when multiple publishers are sending data using UC compared to MC and is counter-intuitive.

Summary of 5Pi Multicast Analysis

Table 4.5 summarises the results of the analysis by showing the most performant communication type (either UC or MC) for each combination of publisher and subscriber count. Similar to the 3Pi analysis, MC outperforms UC in 10 cases for latency and 9 cases for throughput while UC only outperforms MC in 2 cases for throughput and none for latency. Similar results were seen in terms of latency for all cases with 1 and 5 publishers and n subscribers and 10 publishers and subscribers when compared to the 3Pi analysis. This demonstrates a trend that MC generally outperforms UC in these configurations across different physical testbeds. What is particularly interesting is that MC also outperforms UC in the case with 1 publisher and 1 subscriber for both latency and throughput which was also seen in the 3Pi analysis. This is an unexpected result as one would expect both communication types to perform similarly in this simple scenario.

4.3.5 Analysis on the 5 Raspberry Pi Quality of Service Dataset

The subsection analyses the 5Pi QoS Capture dataset, and as detailed in Subsection 3.4.5, includes results from experiments with different data lengths (100 B instead of 512 B), reliability

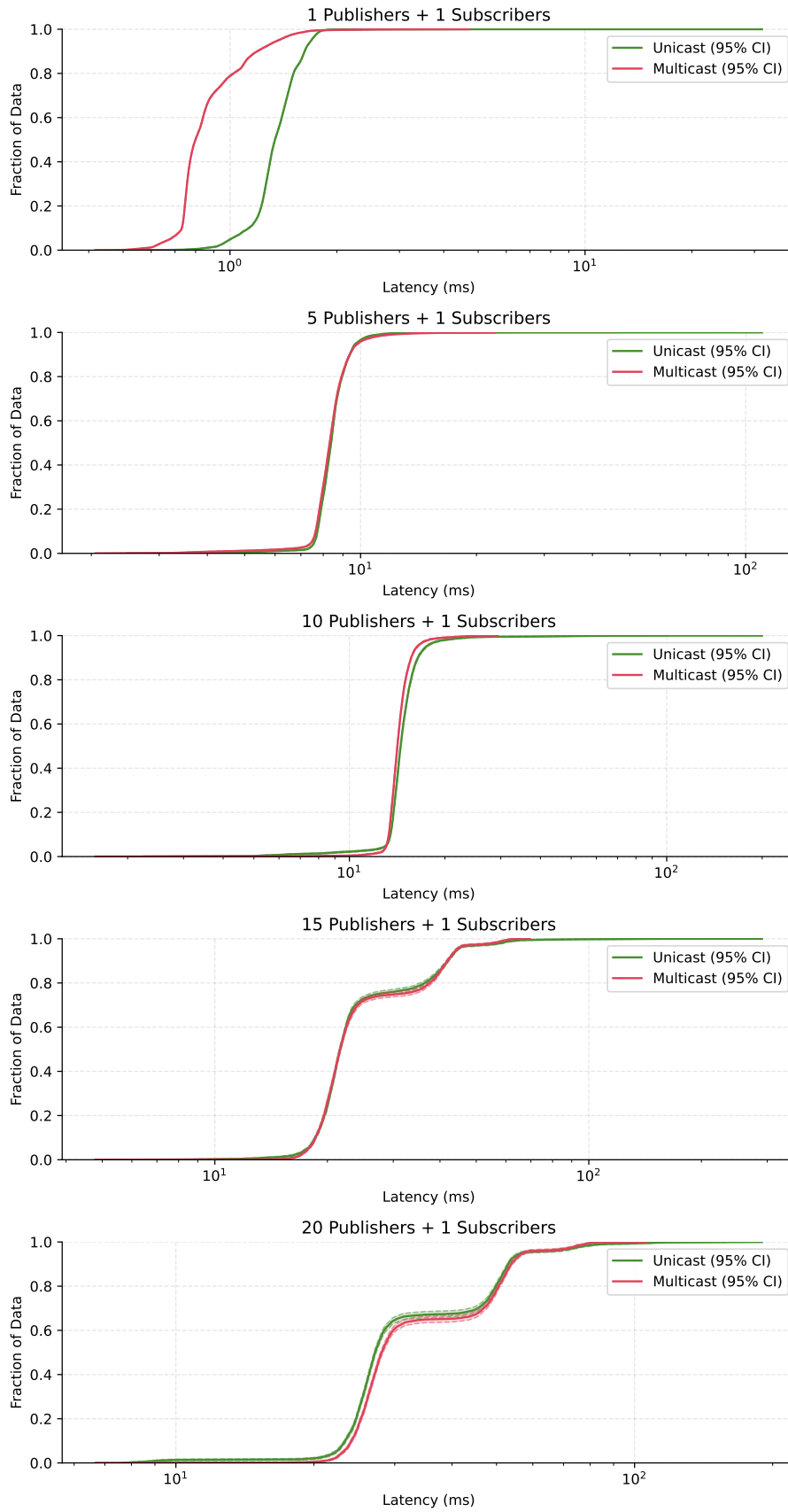


Figure 4.25: eCDFs of MC (red) and UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 20 publishers for the 5Pi MC dataset. The x-axis is in log scale.

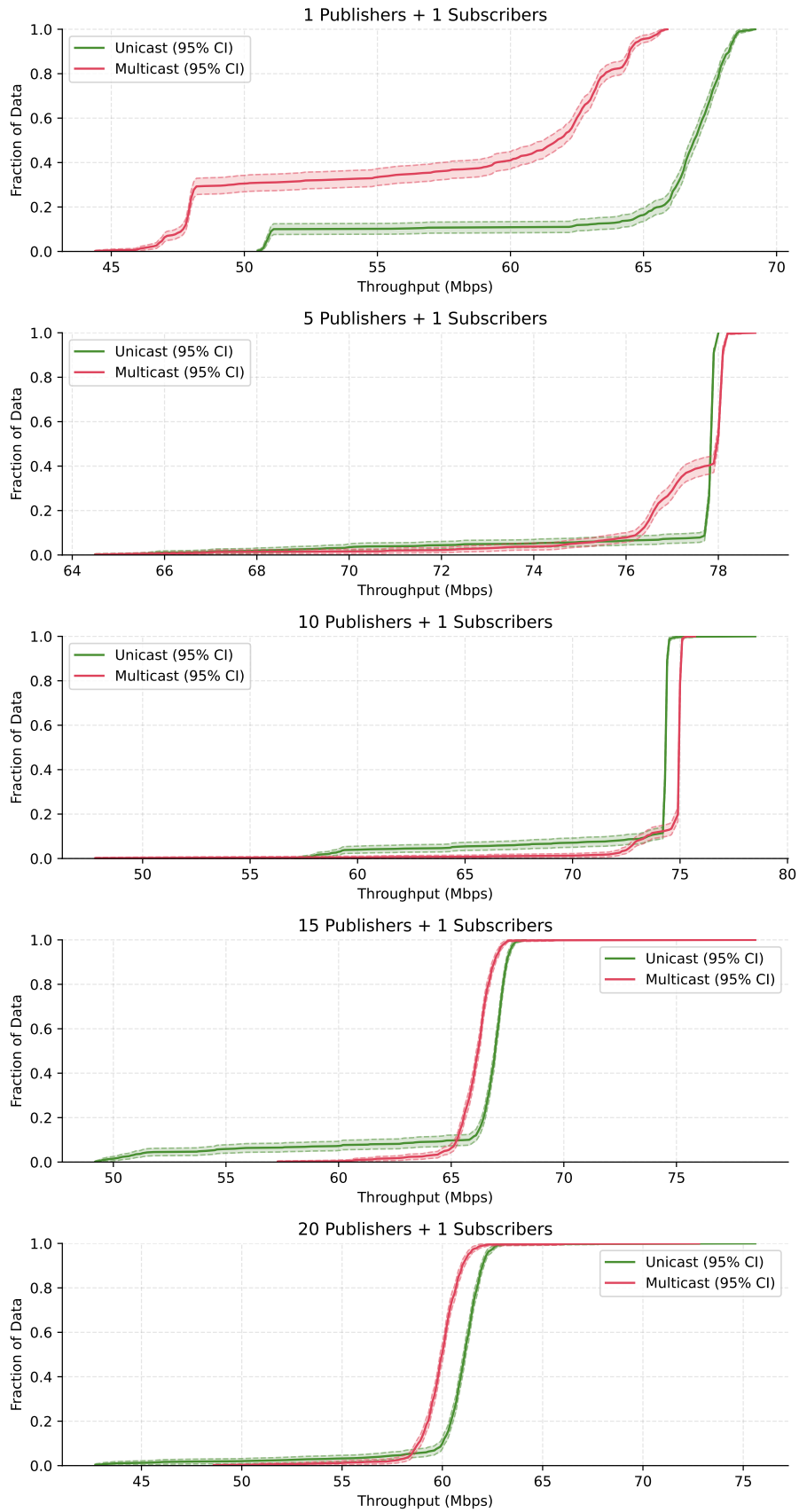


Figure 4.26: eCDFs of MC (red) and UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 20 publishers for the 5Pi MC dataset.

Publisher Count	Subscriber Count	Latency	Throughput
1	1	MC	UC
1	5	MC	mostly MC
1	10	MC	MC
1	15	MC	MC
1	20	MC	MC
1	25	MC	MC
1	50	MC	MC
5	1	none	none
5	5	MC	mostly MC
10	1	none	none
10	10	MC	MC
15	1	none	none
15	15	mostly MC	MC
20	1	none	mostly UC
Total	UC	0	2
	MC	10	9
	none	4	3

Table 4.5: Comparison of Latency and Throughput for Different Publisher and Subscriber Counts. Most performant communication type (UC or MC) per participant count determined by stochastic ordering seen in eCDF. “Mostly MC” indicates that the majority of the eCDF showed stochastic ordering in favour of MC with some intersection. Results also seen from the 3Pi analysis as well as here are highlighted in bold. “none” indicates that there was no clear difference between the 2 communication types.

(both reliable and best effort instead of just reliable) and durability settings (all levels instead of just level 0). This introduces new variables that may impact the performance of MC versus UC communication and the analysis aims to uncover any differences in performance under these varied QoS conditions.

Just like in the previous analyses, we break down the results into 3 sections (where $n > 1$):

1. Analysis of n Publishers and n Subscribers
2. Analysis of 1 Publisher and n Subscribers
3. Analysis of n Publishers and 1 Subscriber

Analysis of n Publishers and n Subscribers

Latency Analysis of the eCDFs depicting latency performance in Figure 4.27 reveals similar performance for both communication types in the case with 5 publishers and subscribers. MC outperforms UC in the cases with 1 and 10 publishers and subscribers with clear stochastic ordering except for some intersection at higher percentiles in the case with 1 publisher and subscriber. This result aligns with the findings from the 5Pi MC exploration dataset, though MC outperformed UC in the 5-participant case there. The step-like pattern seen in the case of 5 publishers and 5 subscribers may be attributed to the inclusion of other QoS parameters

used in this campaign, namely a smaller message size (100 B versus 512 B), the inclusion of best-effort reliability, and the testing of all durability levels. These variations in QoS evidently play a role in shaping the latency performance characteristics observed.

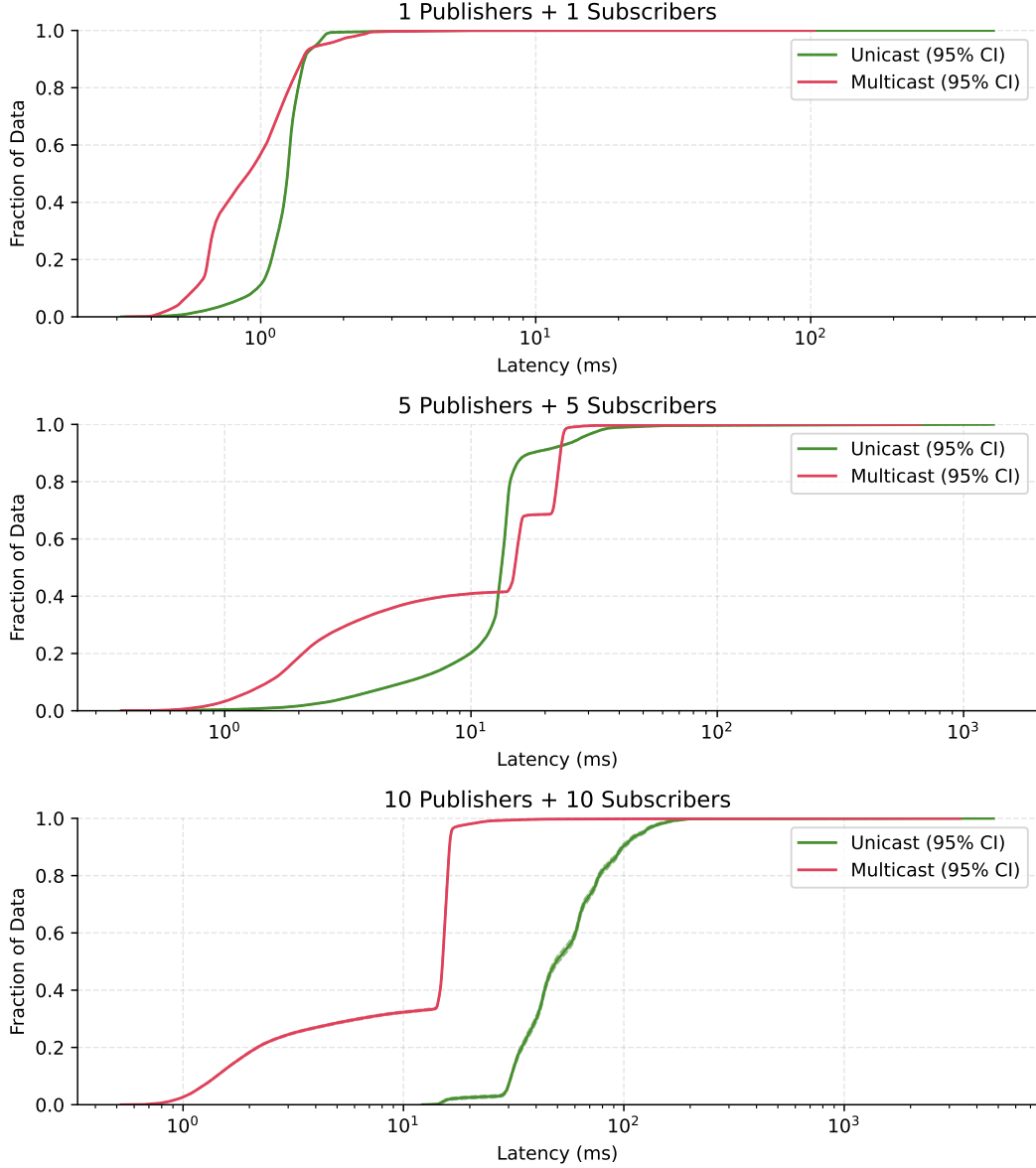


Figure 4.27: eCDFs of MC (red) and UC (green) latency for 1, 5, and 10 publishers and subscribers for the 5Pi QoS dataset. The x-axis is in log scale.

Throughput For the single publisher and subscriber scenario, the throughput performance is broadly similar for both communication types with many overlapping values as shown in Figure 4.28. However, for configurations with 5 and 10 participants, MC demonstrates superior throughput, which aligns with theoretical expectations. This observation is consistent with the results from the 5Pi MC exploration dataset, indicating that the performance advantage of MC persists even with different QoS parameters (smaller data length, best effort communication, and all durability levels) for throughput. The eCDFs for the 1-participant case exhibit a distinct

step-like pattern for both communication types. This pattern suggests the presence of distinct performance clusters within the aggregated data, likely corresponding to different individual experiments within the campaign. This observation highlights the need for a more granular analysis at the level of individual experiments to avoid the confounding effects of aggregation. A closer inspection of the vertical segments within the stepped eCDF suggests that for individual experimental configurations, MC may consistently outperform UC.

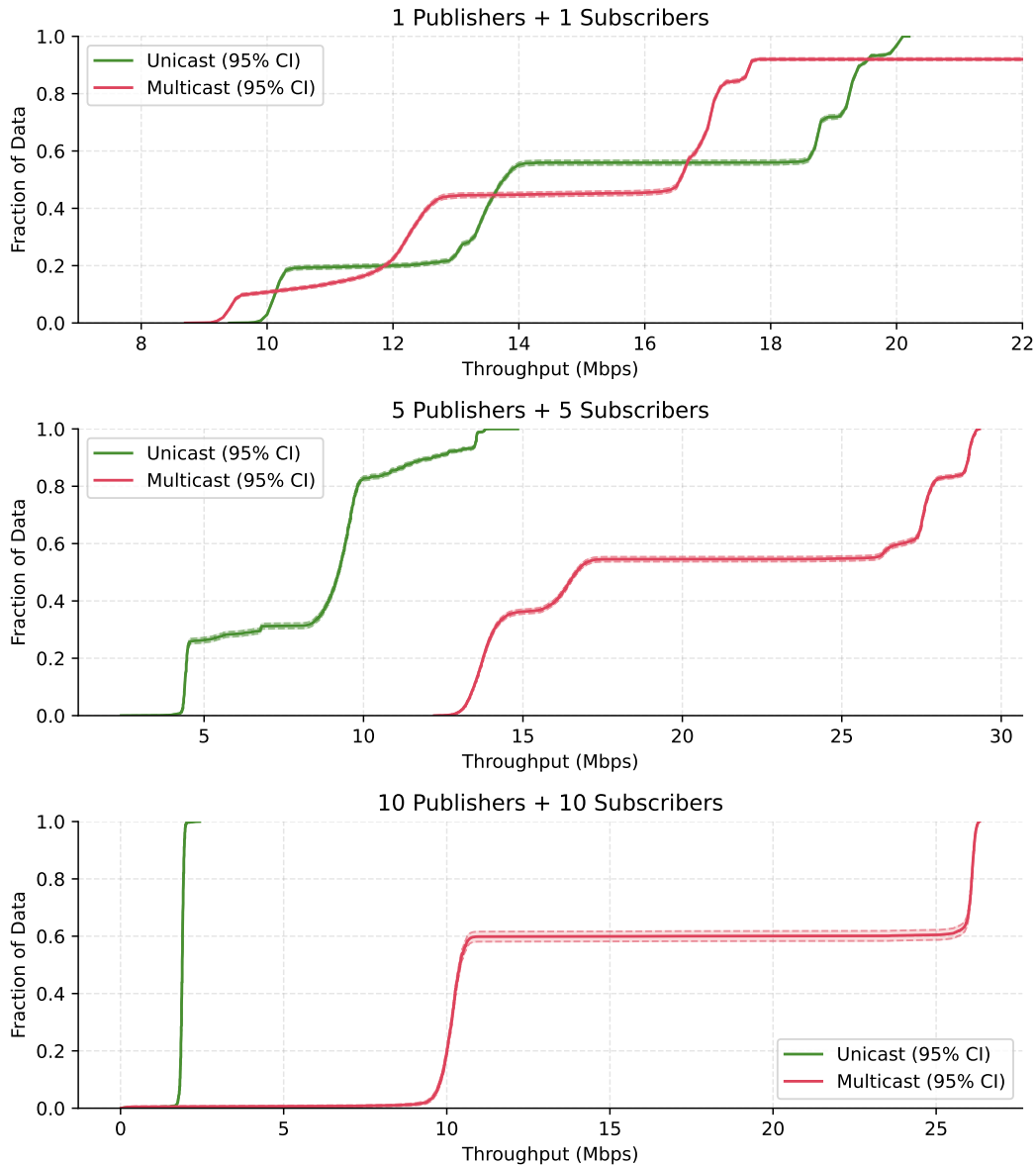


Figure 4.28: eCDFs of MC (red) and UC (green) throughput for 1, 5, and 10 publishers and subscribers for the 5Pi QoS dataset.

Analysis of 1 Publishers and n Subscribers

Latency The eCDFs in Figure 4.29 show noticeable crossover points in all cases (1 publisher with 1, 5, 10, and 25 subscribers). These crossover points seem to shift downwards as the

number of subscribers increases. This demonstrates that MC's performance advantage increases as the system scales, supporting the hypothesis of its efficiency in 1-to-n scenarios. These results show that in the case with 5 and 10 subscribers, MC outperforms UC and especially so at higher percentiles. With the added QoS parameters (smaller data length, best effort communication, and all durability levels), these results further demonstrate the performance advantage of MC in 1-to-n scenarios in terms of latency.

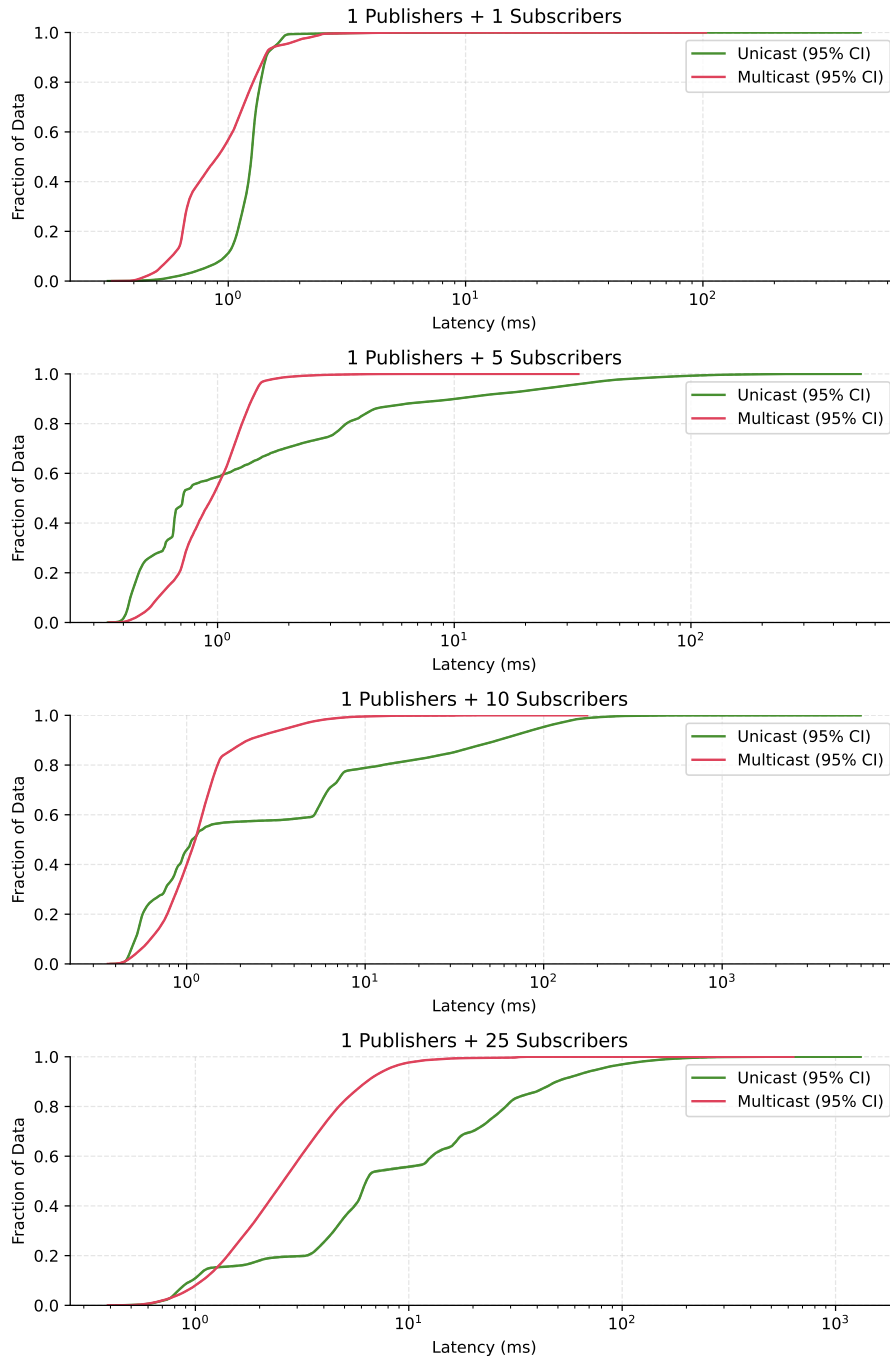


Figure 4.29: eCDFs of MC (red) and UC (green) latency for 1 publisher and 1, 5, 10, and 25 subscribers for the 5Pi QoS dataset. The x-axis is in log scale.

Throughput The eCDFs in Figure 4.30 depicting the throughput results further support the latency findings. Here, MC consistently outperforms UC across all subscriber counts (1, 5, 10, and 25).

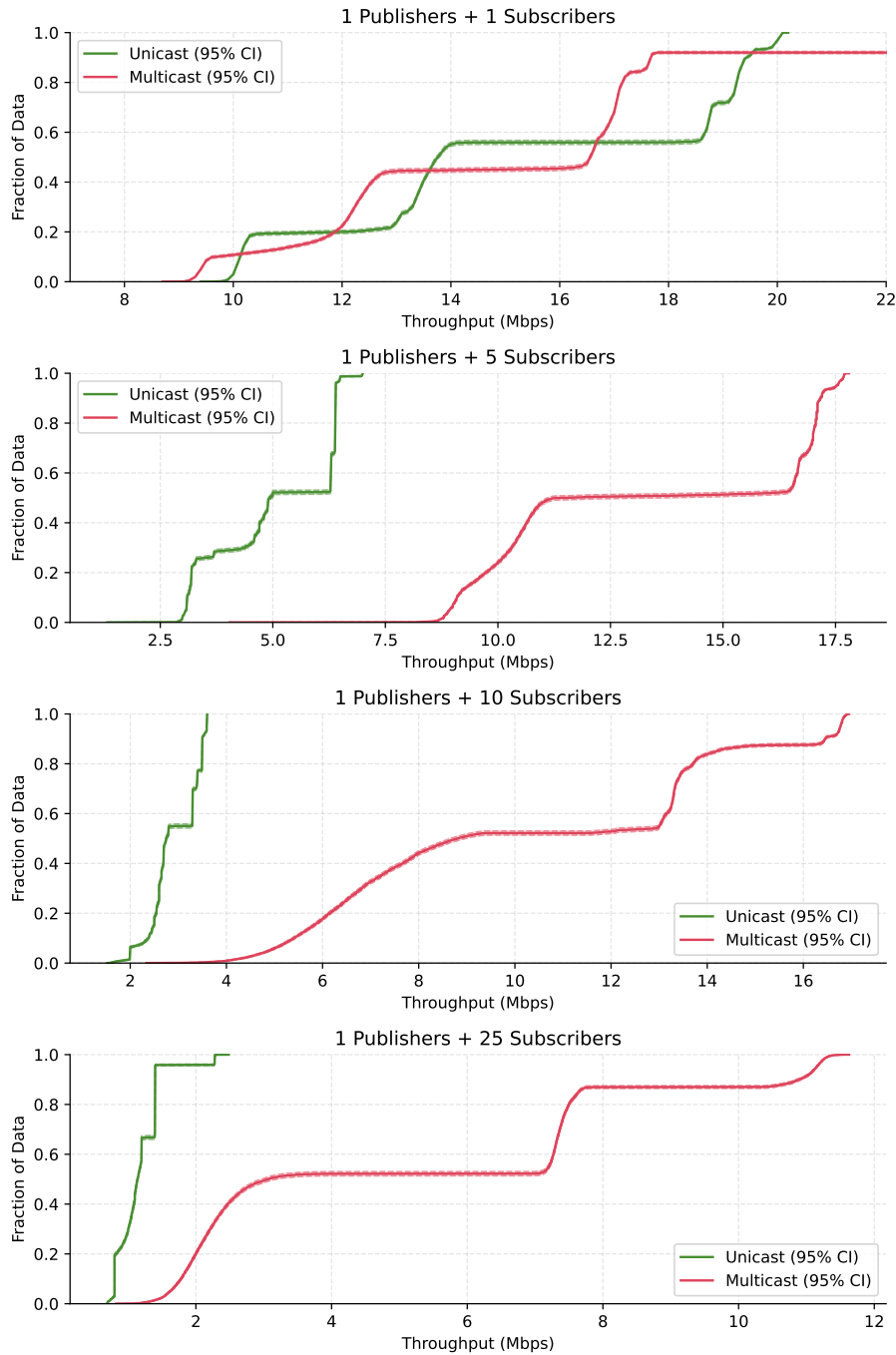


Figure 4.30: eCDFs of MC (red) and UC (green) throughput for 1 publisher and 1, 5, 10, and 25 subscribers for the 5Pi QoS dataset.

Analysis of n Publishers and 1 Subscriber

Latency The eCDFs shown in Figure 4.31, are very similar to those seen in the 5Pi MC exploration dataset. In all cases (5, 10, and 25 publishers with 1 subscriber), both communi-

cation types exhibit nearly identical performance with a slight gap seen in the first half of the percentile range which increases with the number of publishers. Whilst this gap is not sufficient to establish clear stochastic ordering in favour of either communication type for 5 and 10 publishers, it is consistent across these two configurations and indicates a minor performance edge for MC. Much like the previous 5Pi MC exploration dataset, noticeable step-like patterns emerge and a more granular analysis at the level of individual experiments would be required to avoid the confounding effects of aggregation.

Throughput The eCDFs in Figure 4.32 depict throughput results that are very similar to those seen in the 5Pi MC exploration dataset. Similar performance is observed between both communication types for 5 and 10 publishers with 1 subscriber (just like in the case with the 5Pi MC exploration dataset). A surprisingly similar pattern is seen in the case of 25 publishers here and 20 publishers seen in the 5Pi MC exploration dataset where UC has a more vertical shape while MC has a more pronounced single step around the 50th percentile. This repeated observation across different datasets suggests that there may be underlying network dynamics or QoS configurations that consistently influence throughput performance in n-to-1 scenarios.

Summary of 5Pi Quality of Service Analysis

Table 4.6 summarises the results of the analysis by showing the most performant communication type (either UC or MC) for each combination of publisher and subscriber count. A majority of the cases for latency show no clear difference between UC and MC (6 out of 11) while MC outperforms UC in 4 cases and UC outperforms MC in 1 case. For throughput, MC outperforms UC in 8 cases while UC outperforms MC in none and 3 cases show no clear difference between the 2 communication types. These results indicate that MC generally outperforms UC in terms of throughput in the 5Pi QoS dataset. However, throughout the results many instances of step-like patterns are seen in both latency and throughput eCDFs which suggest that a more granular analysis at the level of individual experiments would be required to avoid the confounding effects of aggregation introduced by the varied QoS parameters (smaller data length, best effort communication, and all durability levels).

4.3.6 Comparative Analysis of Latency Across Testbeds

We analysed the impact of MC on latency and throughput performance across virtualised (VMs) and physical (3Pi and 5Pi) testbeds. In this subsection, we focus on latency results across all

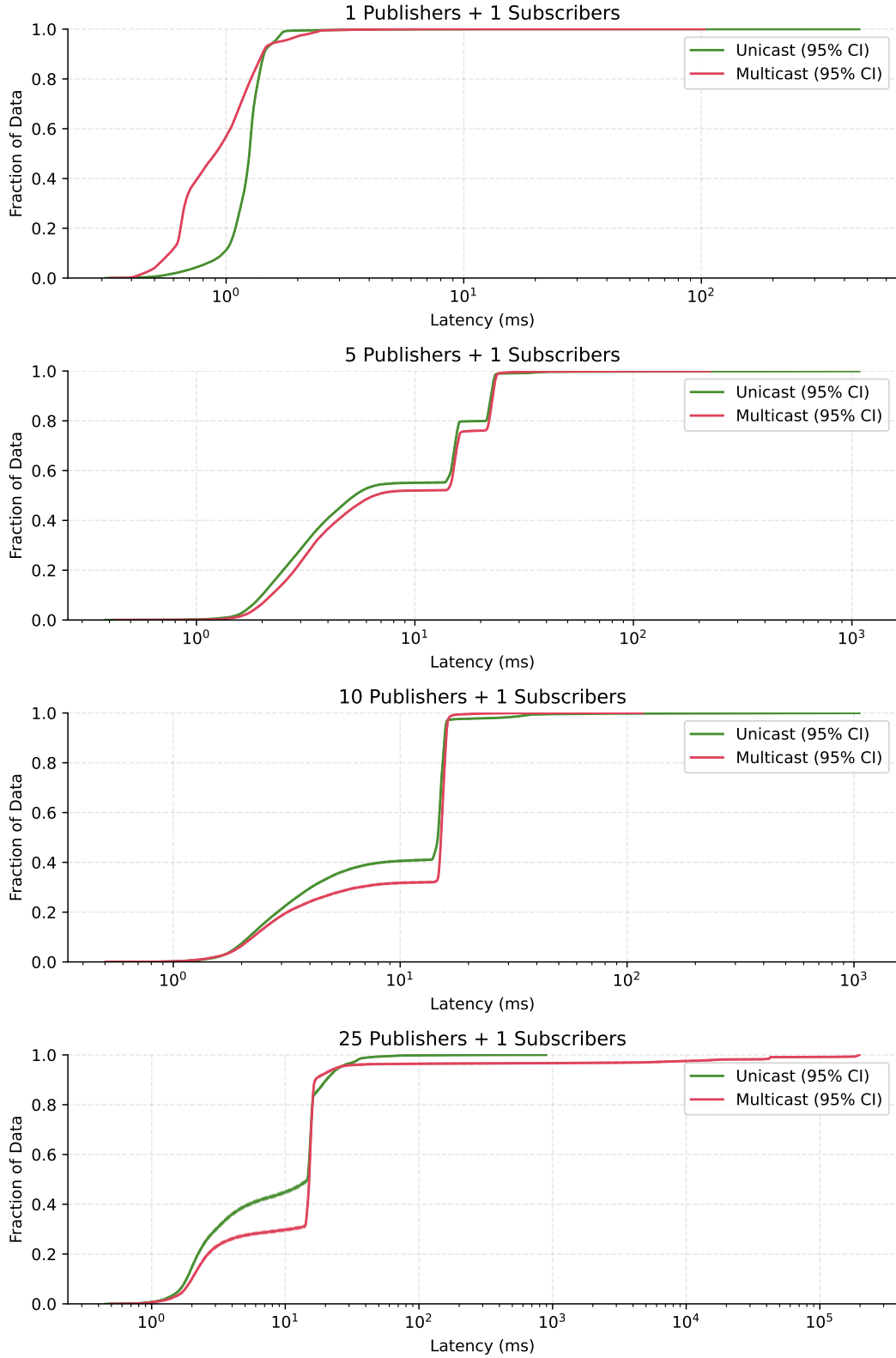


Figure 4.31: eCDFs of MC (red) and UC (green) latency for 1 subscriber and 1, 5, 10, and 25 publishers for the 5Pi QoS dataset. The x-axis is in log scale.

datasets to identify consistent performance trends, specifically on physical testbeds because they provide a more “cleaner” insight. Our findings are summarised in Table 4.7. The table shows almost exact results between the 3Pi and 5Pi testbeds with the MC exploration dataset

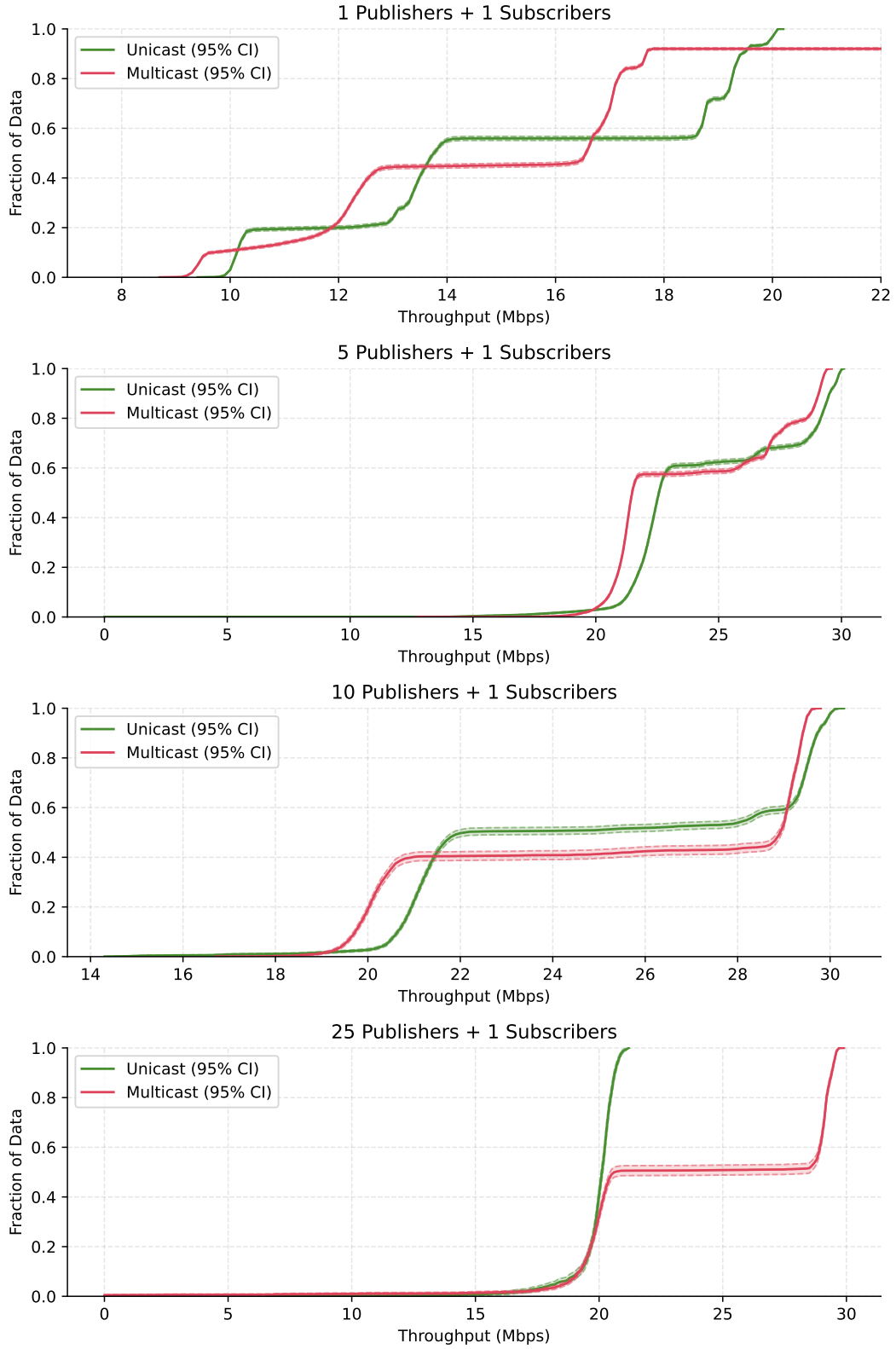


Figure 4.32: eCDFs of MC (red) and UC (green) throughput for 1 subscriber and 1, 5, 10, and 25 publishers for the 5Pi QoS dataset.

where UC never outperforms MC and MC outperforms UC in 10 cases for both testbeds.

When comparing the results for the MC dataset between the two testbeds, there are two similarities. Firstly, MC is more performant in all cases with 1 publisher and n subscribers (1, 5,

Publisher Count	Subscriber Count	Latency	Throughput
1	1	mostly MC	none
1	5	none	MC
1	10	none	MC
1	25	mostly MC	MC
5	1	mostly UC	none
5	5	none	MC
5	10	none	MC
5	25	MC	MC
10	1	none	none
10	10	MC	MC
25	1	none	mostly MC
Total	UC	1	0
	MC	4	8
	none	6	3

Table 4.6: Comparison of Latency and Throughput for Different Publisher and Subscriber Counts for the 5Pi QoS dataset. Most performant communication type (UC or MC) per participant count determined by stochastic ordering seen in eCDF. “Mostly MC” indicates that the majority of the eCDF showed stochastic ordering in favour of MC with some intersection.

10, 15, and 25 subscribers) for both testbeds. This demonstrates the superiority of MC in 1-to-n communication scenarios across different physical testbeds. However, this trend is not seen with the introduction of varied QoS parameters (smaller data length, best effort communication, and all durability levels). This indicates that the introduction of new parameter values are likely interacting with the communication type to influence performance and exploration investigating these interactions would be a valuable avenue for future work. Secondly, MC is more performant in the cases with 1, 5, and 10 publishers and subscribers for both testbeds, though this trend is not seen with 15 or 25 publishers and subscribers which may indicate a limit to how well MC scales with more physical machines. A difference seen in the results between the two testbeds is that MC is more performant with 10 and 15 publishers and 1 subscriber on the 3Pi testbed but not on the 5Pi testbed. This may hint at two possibilities: either MC performs better with less physical machines or MC performs worse with more physical machines. Though this is only seen with 1 subscriber and further investigation would be required to confirm either hypothesis.

When comparing the outcomes across all datasets, the only consistent results show MC outperforming UC in the cases with 1 publisher and 1 subscriber, 10 publishers and 10 subscribers, and 1 publisher and 25 subscribers. This recurrence across all datasets and testbeds highlights specific configurations where MC consistently excels in latency performance and further investigation can reveal if this is truly coincidental or if there are underlying factors at play. The aim of this analysis is to objectively present the comparative performance of MC and UC across

different testbeds without delving into speculative explanations for the observed trends. Future work involves exploring the underlying causes of these performance patterns in greater detail including investigating the interactions between QoS parameters and communication types.

Publishers	Subscribers	3Pi	5Pi	5Pi QoS
1	1	MC	MC	mostly MC
1	5	MC	MC	none
1	10	MC	MC	none
1	15	MC	MC	-
1	25	MC	MC	mostly MC
1	50	-	MC	-
5	1	none	none	mostly UC
5	5	MC	MC	none
5	10	-	-	none
5	25	-	-	MC
10	1	MC	none	none
10	10	MC	MC	MC
15	1	MC	none	-
15	15	none	mostly MC	-
20	1	-	none	-
25	1	MC	-	none
25	25	none	-	-
Total	UC	0	0	1
	MC	10	10	4
	none	3	4	6

Table 4.7: Most performant communication type (UC or MC) according to latency for each variation of publisher and subscriber count summarised across all datasets on physical testbeds. Values in bold represent results consistent across all testbeds. “-” indicates that the configuration was not tested in that dataset.

4.3.7 Comparative Analysis of Throughput Across Testbeds

Just like in the previous subsection, we analysed the impact of MC on throughput performance across virtualised (VMs) and physical (3Pi and 5Pi) testbeds. In this subsection, we focus on throughput results across all datasets to identify consistent performance trends, specifically on physical testbeds because they provide a more “cleaner” insight. Our findings are summarised in Table 4.8 and shows a clearer trend that holds consistent across all datasets (unlike latency results). Overall, MC outperforms UC in most configurations (many of which are consistent across testbeds) while UC rarely outperforms MC. There are more configurations where both communication types show similar performance than configurations where UC outperforms MC.

When comparing between 3Pi and 5Pi for the MC dataset, there are three similarities. Firstly, MC is more performant in all cases with 1 publisher and 5, 10, and 15 subscribers

for both testbeds which demonstrates the superiority of MC in 1-to-n scenarios. Secondly, MC is more performant in the cases with 5, 10, and 15 publishers and subscribers. This consistent pattern across both testbeds highlights the scalability of MC in n-to-n communication scenarios. Finally, both testbeds show similar performance for the case with 5 publishers and 1 subscriber which is unexpected as UC should perform worse in this scenario due to each publisher sending many messages to the single subscriber. These similarities highlight two points: the effectiveness of MC in 1-to-n and n-to-n scenarios across different physical testbeds, and the need for further investigation into the unexpected performance parity in the 5 publishers and 1 subscriber case.

Whilst there are many similarities between the two testbeds, there are more differences. Firstly, in the case with 1 publisher and 1 subscriber, UC is more performant on the 5Pi testbed while both communication types show similar performance on the 3Pi testbed. While these testbeds have different numbers of machines, for this configuration both testbeds use 2 machines (1 per participant). Therefore, the difference in performance may be attributed to the hardware specifications of the machines used in each testbed. Another difference is seen in the case with 1 publisher and 25 subscribers where UC is more performant on 3Pi while MC is more performant on 5Pi which could mean two things: either UC performs better with less physical machines or MC performs better with more physical machines. This difference is only seen in this configuration and further investigation would be required to confirm either hypothesis. The third difference observed is in the case with 10 publishers and 1 subscriber where MC is more performant on 3Pi while both communication types show similar performance on 5Pi. This similarity in performance seen on 5Pi is unexpected because with more publishers and only 1 subscriber, UC should perform worse due to each publisher sending many messages to the single subscriber. One could speculate that the load distribution on the 2 slave machines (3Pi) is better here than the load distribution on the 4 slave machines (5Pi) but further investigation would be required to confirm this. Finally, a difference is seen in the case with 15 publishers and 1 subscriber where MC is more performant on 3Pi while both communication types show similar performance on 5Pi. This difference also sparks the same questions as the previous difference and further investigation would be required to confirm any hypotheses.

Whilst there are only three similarities seen across all datasets, this consistency may show insight into how MC performs in certain configurations regardless of the testbed or QoS parameters used. The first similarity is seen in the case with 1 publisher 5 and 10 subscribers where MC is more performant across all testbeds. Much like previous discussions, this demonstrates the superiority of MC in 1-to-n scenarios. The second similarity is seen in the case with 5

publishers and 1 subscriber where both communication types show similar performance across all testbeds. This unexpected result once again highlights the need for further investigation into this configuration because UC should perform worse in this scenario due to each publisher sending many messages to the single subscriber. Finally, in the cases with 5 and 10 publishers and subscribers, MC is more performant across all testbeds which highlights the scalability of MC in n-to-n communication scenarios. Overall, these consistent results across all datasets and testbeds highlight specific configurations where MC consistently excels in throughput performance and further investigation can reveal if this is truly coincidental or if there are underlying factors at play.

While similarities point to recurring behaviour regardless of testbed or QoS parameters, differences highlight areas where performance varies based on these factors and can provide insight into the conditions under which each communication type excels or falters. The first difference is seen in the case of 1 publisher and 1 subscriber where UC is more performant with the MC dataset on 5Pi while all other datasets show similar performance between both communication types. The interesting aspect of this difference is the consistency seen across the 3Pi and 5Pi QoS datasets where both communication types show similar performance. This deviation in the outcome for the 5Pi MC dataset has to be further investigated to understand why UC performs better in this simple configuration only in this dataset. The second difference is seen in the case with 1 publisher and 25 subscribers where UC is more performant on 3Pi while MC is more performant on the 5Pi (for both datasets). This difference may highlight a case where MC scales better with more physical machines or UC performs better with less physical machines. Further investigation would be required to confirm either hypothesis. The final difference is seen in the case with 10 publishers and 1 subscriber where MC is more performant on 3Pi while both communication types show similar performance on 5Pi (for both datasets). It is strange that MC is more performant in this case on 3Pi and this may be due to the load distribution on the 2 slave machines (3Pi) being better here than the load distribution on the 4 slave machines (5Pi and 5Pi qos).

As mentioned before, the aim of this analysis is to objectively present the comparative performance of MC and UC. As such, many questions have been raised throughout this subsection regarding the observed differences and future work involves exploring the underlying causes of these performance patterns in greater detail including investigating the interactions between QoS parameters and communication types.

Publishers	Subscribers	3Pi	5Pi	5Pi QoS
1	1	none	UC	none
1	5	MC	mostly MC	MC
1	10	MC	MC	MC
1	15	MC	MC	-
1	25	mostly UC	MC	MC
1	50	-	MC	-
5	1	none	none	none
5	5	MC	mostly MC	MC
5	10	-	-	MC
5	25	-	-	MC
10	1	MC	none	none
10	10	MC	MC	MC
15	1	mostly MC	none	-
15	15	MC	MC	-
20	1	-	mostly UC	-
25	1	MC	-	mostly MC
25	25	mostly MC	-	-
Total	UC	1	2	0
	MC	10	9	8
	none	2	3	3

Table 4.8: Most performant communication type (UC or MC) according to throughput for each variation of publisher and subscriber count summarised across all datasets on physical testbeds. Values in bold represent results consistent across all testbeds. “-” indicates that the configuration was not tested in that dataset.

4.3.8 Quantile-Based Analysis

In previous sections that analysed the impact of MC communication, the results revolved around the number of publishers and subscribers and the eCDFs were aggregated across all experimental configurations within each dataset. This aggregation approach, while useful for identifying general trends, has limitations. A subset of the eCDF plots showed step-like patterns indicating clustering of performance data. This clustering suggests that different experimental configurations may have distinct performance characteristics that are obscured when data is aggregated. To address this limitation, a new approach is proposed that involves categorising individual experimental configurations. This method uses the PAT tool mentioned in Section 3.3.4 to extract the every individual unique experimental configurations from all datasets and compare the performance of MC and UC within these configurations. This allows the identification of specific scenarios where one communication type outperforms the other, providing a more granular understanding of performance dynamics ensuring that the unique characteristics of each configuration are preserved and analysed.

The output of this approach consists of a list individual unique configurations and whether MC or UC or neither (labelled as “Uncategorised”) was the more performant communication type based on latency and throughput metrics. This granularity is further enhanced by analysing performance across different percentile ranges of the latency and throughput distributions. These percentile ranges help to capture performance variations that may occur at different points in the distribution, such as best-case, average-case, and worst-case scenarios. We define these percentile ranges as follows: the 0 to 5th percentile captures the best-case performance, the 5th to 95th percentile represents the average-case performance, the 95th to 100th percentile reflects the worst-case performance, and 0 to 100th percentile encompasses the overall performance.

The quantile-based analysis described in this section is broken down into two main parts: latency and throughput. Each metric is further divided into two types of analysis: distribution and statistical. The distribution analysis focuses on the distributions of the parameter values within the defined percentile ranges, while the statistical analysis employs hypothesis testing to determine the significance of performance differences between MC and UC within these ranges. Therefore, within each type of analysis (distribution or statistical), we present findings for each percentile range (0 to 5th, 5th to 95th, 95th to 100th, and 0 to 100th percentiles).

Distribution Analysis of Latency

A summary of the distribution of cases across the defined percentile ranges is presented in Table 4.9. This table breaks down the number of experimental configurations where UC, MC, or neither (“Uncategorised”) outperformed the other within each percentile range. There may be cases where some configurations show MC outperforming UC in certain percentile ranges while UC outperforms MC in others though this is likely to be rare. This section is further broken down into analyses for each percentile range.

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
UC	12	19	28	2
MC	18	60	114	17
Uncategorised	139	90	27	150
Total	169	169	169	169

Table 4.9: Distribution of cases across percentile ranges for UC and MC communication (“Uncategorised” indicates no stochastic ordering).

0th to 5th Percentile The distribution of experimental configurations for the 0 to 5th percentile is detailed in Table 4.9 and visualised in Figure 4.33. For the 12 cases where UC outperformed MC, experiments were distributed across several data lengths; 8 cases involved 100 B packets, 2 used 1 KB, and 2 employed 512 B. Publisher counts were spread across all values with no clear concentration and subscriber counts were similar with notable peaks at 5 and 25 subscribers. A majority of these cases used reliable communication while durability was spread across levels of 0 (6 cases) and 3 (8 cases).

A total of 18 cases of the 169 showed MC as the most performant communication type. Publisher and subscriber counts for MC were notably lower than for UC in this performance bracket. None of these cases used best effort communication and the durability levels were mostly low.

The remaining 139 uncategorised cases in this percentile were dominated by experiments using a 100 B data length. Publisher and subscriber counts for these cases ranged mostly between 5 and 10 participants. Reliable communication was seen to be twice as common as best-effort and all cases used a durability level of 0.

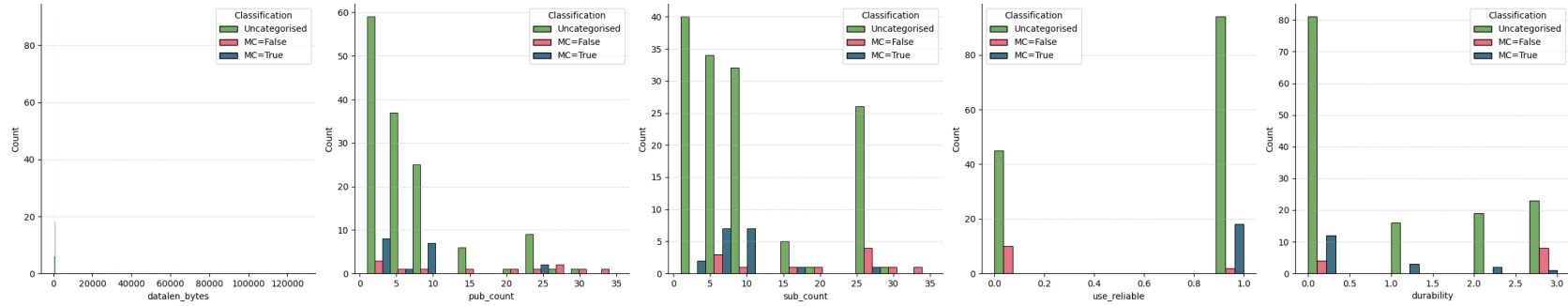


Figure 4.33: Histogram of data distribution for the 0 to 5 percentile range.

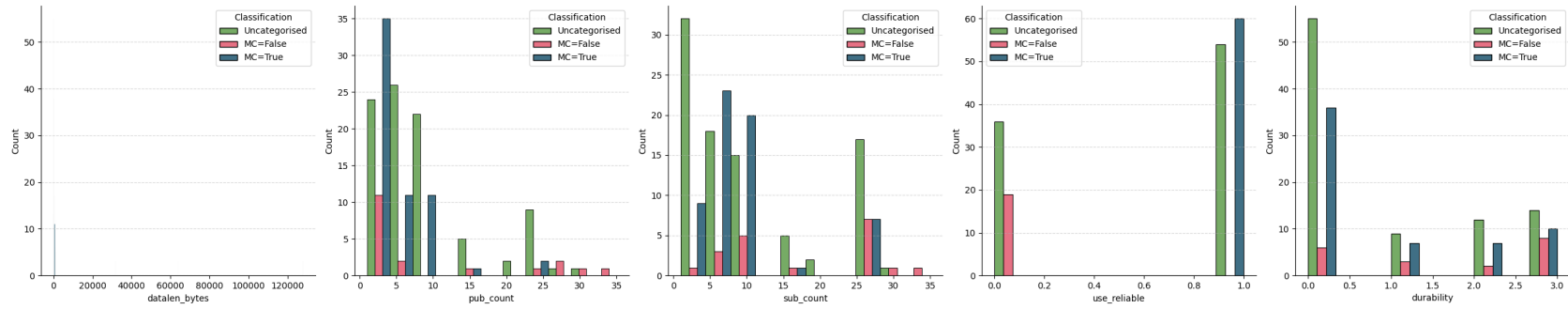


Figure 4.34: Histogram of data distribution for the 5 to 95 percentile range.

5 to 95 Percentile For the 5 to 95th percentile, the data are presented in Table 4.9 and Figure 4.34. UC outperformed MC in 19 configurations. Majority of these cases used 100 B (13 cases) while the rest used 1 KB (6 cases). A majority of the cases employed 1 publisher while subscriber counts were spread between 5, 10, and 25 subscribers with a preference for 25. All communication was best effort and durability levels were spread with slight preferences for 0 and 3.

In contrast, 60 configurations favoured MC communication. Here, the publishers count was predominantly 1 with many cases also using 5 and 10 publishers, and the subscriber count peaked at 5 and 10 subscribers. All 60 cases used reliable communication and durability levels were mostly set to 0.

The remaining cases (90) were uncategorised and used primarily 100 B and 1 KB data lengths. Publisher and subscriber counts mostly ranged between 1 to 15 with a spike at 25 subscribers. Reliable communication (55 cases) was more common than best-effort (35 cases) and all durability levels were 0.

From these findings, it can be inferred that UC communication tends to perform better in scenarios with smaller data lengths and higher subscriber counts, particularly when using best-effort communication. Meanwhile, MC communication appears to be more effective in configurations with lower subscriber counts and a consistent use of reliable communication within this mid-range latency bracket. The identification of parameters that influence performance in this range are starting to emerge, but further analysis is needed to draw definitive conclusions.

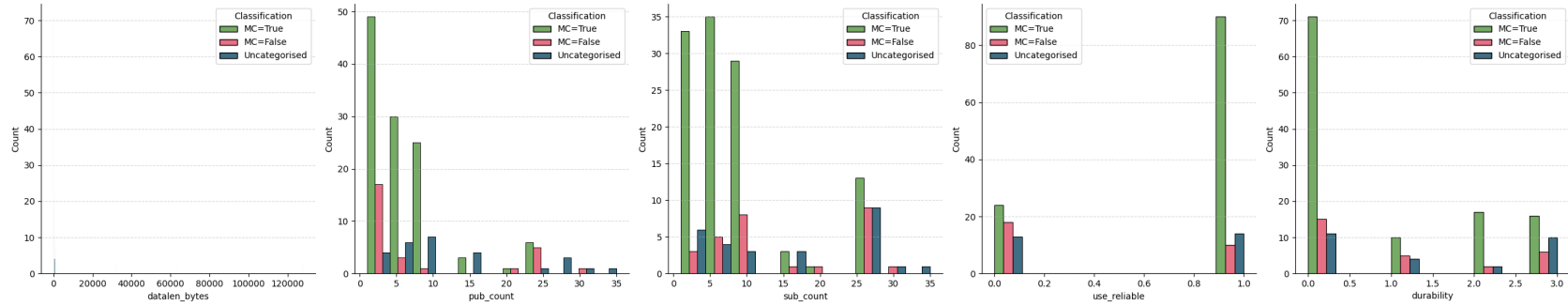


Figure 4.35: Histogram of data distribution for the 95 to 100 percentile range.

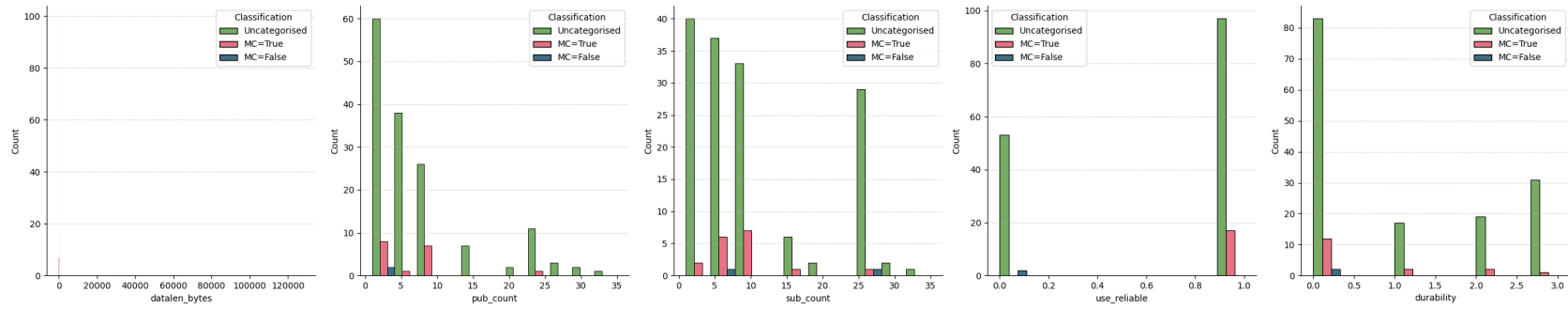


Figure 4.36: Histogram of data distribution for the 0 to 100 percentile range.

95 to 100 Percentile The highest performance bracket is detailed in Table 4.9 and Figure 4.35 and reports 28 configurations where UC communication was more performant, compared to 114 configurations favouring MC. Uncategorised cases were identified in 27 configurations. This indicates a pronounced advantage for MC communication in the higher latency ranges.

The 28 UC cases primarily involved small data lengths, such as 100 B and 1 KB. Publisher counts were low (1 to 5), while subscriber counts were at medium (5 to 10) levels with a spike at 25. Best-effort communication (18 cases) was more common than reliable communication (10 cases) and most of the cases were configured with a durability of 0.

The 114 MC cases showed a strong prevalence for 100 B data lengths (62 cases) and 512 B (21 cases). Publisher counts remained low, and subscriber counts ranged from low to medium levels. Reliable communication and a durability of 0 were the predominant values observed.

100 B was the most common in both the UC and MC cases, which makes it hard to distinguish the impact of data length on performance in this percentile. Therefore, when comparing the second most common data lengths, UC communication had a significant number of cases with 1 KB packets, while MC had a notable presence of 512 B packets. This suggests that UC communication may have an advantage with slightly larger data lengths in this high latency range. Low publisher counts were consistent across both communication types, indicating that the number of publishers may not be a significant factor in this percentile. Subscriber counts were more varied in UC cases, spanning medium levels, while MC cases included both low and medium levels. Best-effort communication was more prevalent in UC cases, whereas reliable communication dominated in MC cases.

The uncategorised cases in this percentile were composed of experiments with 100 B, 1 KB, and 512 B data lengths. Publisher counts ranged from low to medium levels while subscriber counts included both low and high values. Reliability was equally distributed between best-effort and reliable methods and the durability settings included both 0 and 3. These findings suggest that in the highest latency range, UC communication tends to perform better with slightly larger data lengths and a more varied subscriber count, particularly when using best-effort communication. Meanwhile, MC communication appears to be more effective with smaller data lengths and a consistent use of reliable communication.

0 to 100 Percentile The overall distribution is shown in Figure 4.36 where only 2 cases were identified showing UC communication consistently outperforming MC. In contrast, 17 configurations favoured MC communication and the majority of configurations (150) did not show a consistent performance advantage for either communication type across the entire dis-

tribution. For the 17 MC communication across the full percentile range, data lengths were varied, including cases with 100 B and 512 B packets. Publisher counts were consistently low, and subscriber counts ranged from low to medium levels. Reliable communication was the predominantly applied protocol and all cases had a durability setting of 0.

These 150 uncategorised experiments featured a wide array of data lengths, with a high concentration at 100 B. Publisher counts ranged from low to medium, while subscriber counts were mostly low but with a notable spike at 25 subscribers. Reliable communication was more frequently used than best-effort, and all cases were configured with a durability of 0.

The limited number of categorised cases across the full latency distribution suggests that consistent performance advantages for either communication type are rare.

Using all of the insights gained from the individual percentile ranges, a consolidated summary of the key characteristics for each communication type is presented in Table 4.10.

Section	Classification	Cases	Data Length	Publishers	Subscribers	Reliable	Durability
0 to 5	UC	12 of 169	100 B	Spread	Spread (peak at 5 and 25)	Reliable	0 and 3
0 to 5	MC	18 of 169	100 B, 512 B	Low	Low	Reliable	0
5 to 95	UC	19 of 169	100 B	Low	High	Best Effort	Spread
5 to 95	MC	60 of 169	100 B	Low	Low to Mid	Reliable	0
95 to 100	UC	28 of 169	100 B	Low	Mid	Best Effort	0
95 to 100	MC	114 of 169	100 B	Low	Low to Mid	Reliable	0
0 to 100	UC	2 of 169	1 KB	Low	Low and High	Best Effort	0
0 to 100	MC	17 of 169	100 B, 512 B	Low	Low to Mid	Reliable	0

Table 4.10: Summary of case classifications across percentile ranges for latency analysis.

Statistical Analysis of Latency

Solely relying on visual interpretations of histograms can lead to subjective conclusions. To provide a more objective assessment, a statistical analysis was conducted using the χ^2 experiment to evaluate the independence of various parameters on latency performance. This statistical experiment identifies whether there is a significant association between categorical variables and helps to determine which parameters have a meaningful impact on latency outcomes where either MC or UC was more performant. Alongside the χ^2 statistic, the p-value, DoF, and V were calculated to assess the strength of any identified associations. The combination of both the χ^2 statistic and the p-value provides a comprehensive understanding of the relationships between the parameters and latency performance.

To interpret the strength of these associations according to the χ^2 statistic, one can compare the p-value against a significance level (set at 0.05). If the p-value is less than this threshold, it indicates a statistically significant association between the parameter and latency performance. The DoF in the χ^2 test, are calculated as $(r - 1)(c - 1)$, where r and c represent the number of rows and columns in the contingency table, respectively. High DoF, such as 24 for subscriber count and 20 for data length, indicate large contingency tables with multiple categories (e.g., subscriber counts of 1, 5, 10, 25, 50), reflecting complex experimental designs. Moderate DoF, such as 18 for publisher count and 6 for durability, suggest fewer categories, corresponding to varied but less extensive configurations. Low DoF, such as 2 for reliability, imply simple contingency tables (e.g., binary categories like reliable versus best-effort), making the experiment sensitive to small differences. Higher DoF increase the chi-squared test's sensitivity but require larger χ^2 values for statistical significance.

On top of this, the V offers a measure of the strength of the association ranging between 0 and 1, with values closer to 1 indicating a stronger relationship. In summary, a p-value below 0.05 indicates a statistically significant association, and the V quantifies the strength of this association, ranging from 0 to 1. The statistical significance of these parameters is detailed in Tables 4.11 through 4.14.

0 to 5 Percentile Of the 169 cases, 12 configurations favoured UC communication, while 18 configurations favoured MC communication in the 0 to 5 percentile range. According to the χ^2 experiment results in Table 4.11, the parameters with a p-value below the significance threshold of 0.05 are publisher count, subscriber count, reliability, and durability. Of these 4 parameters, publisher count and subscriber count exhibit the highest V values (0.4013 and

Parameter	0 to 5 χ^2	0 to 5 P-Value	0 to 5 DoF	0 to 5 V
Data Length (B)	20.5909	0.4217	20	0.2468
Publisher Count	54.4218	1.6e-05	18	0.4013
Subscriber Count	56.7796	0.0002	24	0.4099
Reliability	22.7863	1.1e-05	2	0.3672
Durability	21.5760	0.0014	6	0.2527

Table 4.11: Statistical analysis for the 0 to 5 percentile range. Bold values indicate statistical significance (p-value < 0.05).

0.4099 respectively), indicating a moderate association with latency performance in this percentile range. Meanwhile, reliability shows a slightly lower V of 0.3672, and durability has the lowest V of 0.2527 among the significant parameters. Data length, with a p-value of 0.4217, does not show a statistically significant association with latency performance in this range. Therefore, the analysis suggests that publisher and subscriber counts are the most influential parameters affecting latency performance in the 0 to 5 percentile range, followed by reliability and durability. This suggests that configurations with varying numbers of publishers and subscribers, as well as the choice of reliability and durability settings, determine latency outcomes in the most performant scenarios.

Parameter	5 to 95 χ^2	5 to 95 P-Value	5 to 95 DoF	5 to 95 V
Data Length (B)	44.2573	0.0014	20	0.3619
Publisher Count	46.2382	0.0003	18	0.3699
Subscriber Count	57.5735	0.0001	24	0.4127
Reliability	70.6080	4.7e-16	2	0.6464
Durability	9.2119	0.1620	6	0.1651

Table 4.12: Statistical analysis for the 5 to 95 percentile range. Bold values indicate statistical significance (p-value < 0.05).

5 to 95 Percentile Of the 169 cases, 19 configurations favoured UC communication, while 60 configurations favoured MC communication in the 5 to 95 percentile range. Between the 5th and 95th percentiles, as shown in Table 4.12, all parameters except durability exhibit p-values below the significance threshold of 0.05. This indicates that data length, publisher count, subscriber count, and reliability have statistically significant associations with latency performance in this range. Reliability stands out with the highest V of 0.6464, suggesting a strong association with latency performance. This is followed by subscriber count and publisher count, with V values of 0.4127 and 0.3699 respectively, indicating moderate associations. Unlike the 0 to 5 percentile range, data length now shows a statistically significant association with

latency performance, albeit with a lower V of 0.3619. Durability, with a p-value of 0.1620, does not show a statistically significant association in this range. Overall, the analysis suggests that reliability is the most influential parameter affecting latency performance in the 5 to 95 percentile range, followed by subscriber and publisher counts. Data length also plays a role, but to a lesser extent, while durability appears to have minimal impact on latency outcomes in this mid-range performance bracket. The change in significance for data length compared to the 0 to 5 percentile range indicates that its impact on latency may be more pronounced in scenarios with a broader range of performance outcomes and this warrants further investigation.

Parameter	95 to 100 χ^2	95 to 100 P-Value	95 to 100 DoF	95 to 100 V
Data Length (B)	22.6879	0.3044	20	0.2591
Publisher Count	55.8326	9.4e-06	18	0.4064
Subscriber Count	54.8032	0.0003	24	0.4027
Reliability	22.7026	1.1e-05	2	0.3666
Durability	11.8186	0.0661	6	0.1870

Table 4.13: Statistical analysis for the 95 to 100 percentile range. Bold values indicate statistical significance (p-value < 0.05).

95 to 100 Percentile Between the 95th to 100th percentiles, 28 configurations favoured UC communication, while 114 configurations favoured MC communication. As detailed in Table 4.13, publisher count, subscriber count, and reliability exhibit p-values below the significance threshold of 0.05. This indicates that these parameters have statistically significant associations with latency performance in this worst-case scenario range. Of these significant parameters, publisher count and subscriber count show the highest V values (0.4064 and 0.4027 respectively), indicating moderate associations with latency performance. Reliability also shows a statistically significant association, with a V of 0.3666, suggesting a moderate relationship. Data length and durability, with p-values of 0.3044 and 0.0661 respectively, do not show statistically significant associations with latency performance in this range. Overall, the analysis suggests that publisher and subscriber counts are the most influential parameters affecting latency performance in the 95 to 100 percentile range, followed by reliability. This indicates that configurations with varying numbers of publishers and subscribers, as well as the choice of reliability settings, play a crucial role in determining latency outcomes in the worst-case scenarios.

0 to 100 Percentile When observing the entire latency distribution from the 0 to 100 percentiles, UC was most performant in 2 cases while MC was most performant in 17. As shown

in Table 4.14, only data length and reliability exhibit p-values below the significance threshold of 0.05. The V values range from 0.2761 for reliability to 0.3207 for data length, indicating moderate associations with latency performance. These values are not as high as those observed in the narrower percentile ranges, suggesting that while these parameters do influence latency performance across the full distribution, their impact is less pronounced when considering all performance levels. This highlights that data length and reliability are the primary factors affecting latency performance when considering the entire range of outcomes, while publisher and subscriber counts do not show statistically significant associations in this broad context. This also signifies that one must consider the specific performance range when evaluating the impact of these parameters on latency, as their influence can vary significantly across different segments of the latency distribution.

Parameter	0 to 100 χ^2	0 to 100 P-Value	0 to 100 DoF	0 to 100 V
Data Length (B)	34.7649	0.0214	20	0.3207
Publisher Count	11.7607	0.8593	18	0.1865
Subscriber Count	11.0219	0.9889	24	0.1806
Reliability	12.8787	0.0016	2	0.2761
Durability	3.8965	0.6907	6	0.1074

Table 4.14: Statistical analysis for the 0 to 100 percentile range.

All V values where the p-value is below 0.05 are summarised in Table 4.15. Only one parameter has a V value above 0.5, which is reliability in the 5 to 95 percentile range and this shows the strongest association with latency performance. Reliability is also the parameter that consistently shows statistical significance across all percentile ranges. Publisher and subscriber counts are significant in three of the four percentile ranges, with V values consistently above 0.4, indicating moderate associations with latency performance. Durability is the least significant parameter, showing statistical significance only in the 0 to 5 percentile range with a V of 0.2527. These results highlight that reliability plays a major role in influencing latency performance across different performance brackets, while publisher and subscriber counts also have notable impacts, particularly in the more extreme percentile ranges.

Distribution Analysis of Throughput

The distribution of cases across these percentile ranges is summarised in Table 4.16 and visualised in Figures 4.37 through 4.40. An inconsistency in the total number of cases for the different percentile ranges is observed with 147 cases in the 0 to 5 percentile, but 167 cases in the other ranges. This discrepancy is due to some experiments producing less samples resulting

Percentile Range	Data Length	Publisher Count	Subscriber Count	Reliability	Durability
0-5	–	0.4013	0.4099	0.3672	0.2527
5-95	0.3619	0.3699	0.4127	0.6464	–
95-100	–	0.4064	0.4027	0.3666	–
0-100	0.3207	–	–	0.2761	–

Table 4.15: Summary of V values for statistically significant parameters across percentile ranges in latency analysis.

in a lack of data to calculate the 0 to 5 percentile. Overall, the table shows that MC communication is more frequently the most performant communication type across all throughput percentile ranges compared to UC communication. This is especially highlighted in the larger throughput percentiles.

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
UC	5	1	18	0
MC	53	109	131	60
Uncategorised	89	57	18	107
Total	147	167	167	167

Table 4.16: Distribution of cases across percentile ranges for MC communication in throughput analysis.

0 to 5 Percentile The distribution for the lowest throughput percentile is shown in Figure 4.37. Of the 147 cases, only 5 cases showed UC outperforming MC, compared to 53 configurations favouring MC while the rest of the cases (89) were uncategorised.

All 5 cases where UC communication was more performant involved a 100 B data length. Publisher counts for these UC experiments were either 5 or 10, while the subscriber count was fixed at 5 and both the reliability and durability settings were evenly distributed among these cases.

The 53 cases where MC was more performant in this percentile was represented by a more diverse set of configurations. The data lengths included 27 cases of 100 B and 9 cases of 512 B, among others. Both publisher and subscriber counts were low (in the range of 1 to 10). Reliable communication (36 cases) was more common than best-effort (16 cases) and all MC experiments were configured with no durability.

The 89 uncategorised cases were predominantly composed of experiments with 100 B data lengths. Publisher counts were generally low but featured a spike at 25, a pattern also seen in the subscriber counts. Reliable communication was significantly more prevalent than best-effort, and all cases used a durability setting of 0.

Data lengths across the 3 classifications were varied, with a strong emphasis on 100 B packets due to the large number of experiments using this size. In terms of publisher and subscriber count, both UC and MC cases were concentrated in the low range (1 to 10) while uncategorised cases showed a broader distribution with a notable peak at 25. Reliable communication was more commonly used across all classifications, particularly in MC and uncategorised cases though UC cases had an even split between reliable and best-effort. This change in the ratio across classifications suggests that reliability may play a role in throughput performance for UC communication. Further investigation using statistical analysis is required to draw definitive conclusions about the impact of these parameters on throughput performance in this lowest percentile range.

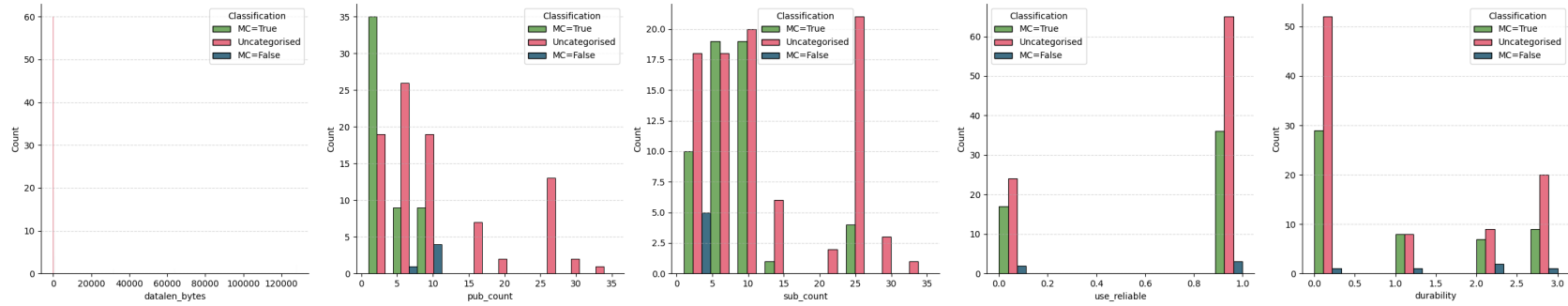


Figure 4.37: Histogram of data distribution for the 0 to 5 percentile range in throughput analysis.

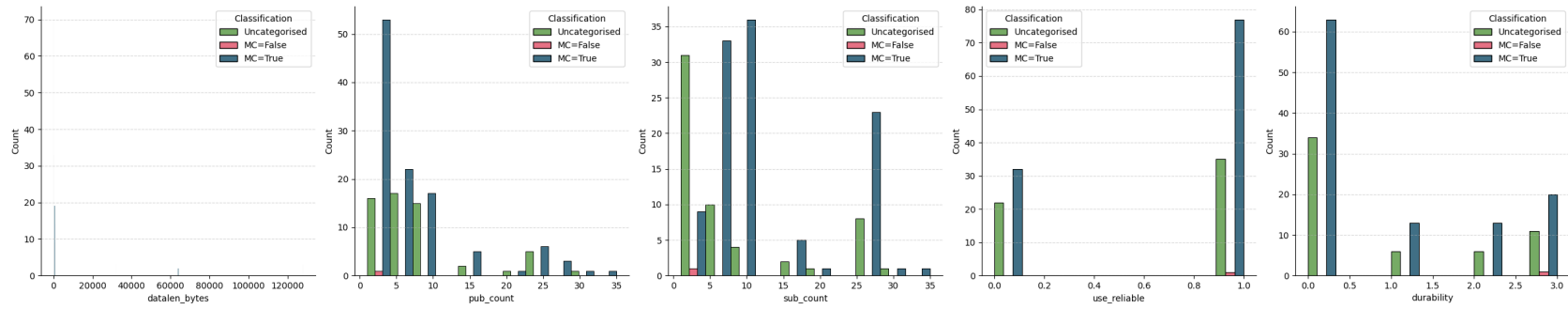


Figure 4.38: Histogram of data distribution for the 5 to 95 percentile range in throughput analysis.

5 to 95 Percentile The histograms of parameter values for the 5 to 95th percentile are and Figure 4.38. Of the 167 cases in this range, only a single configuration was identified where UC communication consistently outperformed MC, compared to 109 configurations favouring MC. Meanwhile, 57 configurations did not show a consistent performance advantage.

MC communication was represented by 109 cases with varied data lengths, although a majority used 100 B packets. Publisher counts were low, while subscriber counts ranged from low to medium levels, with a notable peak at 25. Reliable communication was used in 77 of these cases, compared to 32 for best-effort and all MC experiments had a durability setting of 0.

The 57 uncategorised cases consisted mainly of experiments with 100 B data lengths and low publisher and subscriber counts though a data length of 1 KB was seen in 14 cases. Reliable communication with 35 cases was more common than best-effort (22 cases), and all cases were configured with a durability of 0.

Given the single case where UC communication was more performant, it is challenging to draw definitive conclusions about the impact of specific parameters on throughput performance for UC in this percentile range. However, a comparison between the MC and uncategorised cases may provide insights into the factors that contribute to better throughput performance. Data lengths were varied across both classifications, with a strong emphasis on 100 B packets due to the large number of experiments using this size. Publisher counts were consistently low across both classifications, while subscriber counts in MC cases ranged from low to medium levels, with a notable peak at 25. The ratio of reliable to best-effort communication was similar in both classifications, with reliable being more prevalent. These observations indicate only slight differences in the configurations that lead to better throughput performance in MC communication compared to uncategorised cases. Further statistical analysis is required to determine the significance of these parameters in influencing throughput performance in this mid-range percentile.

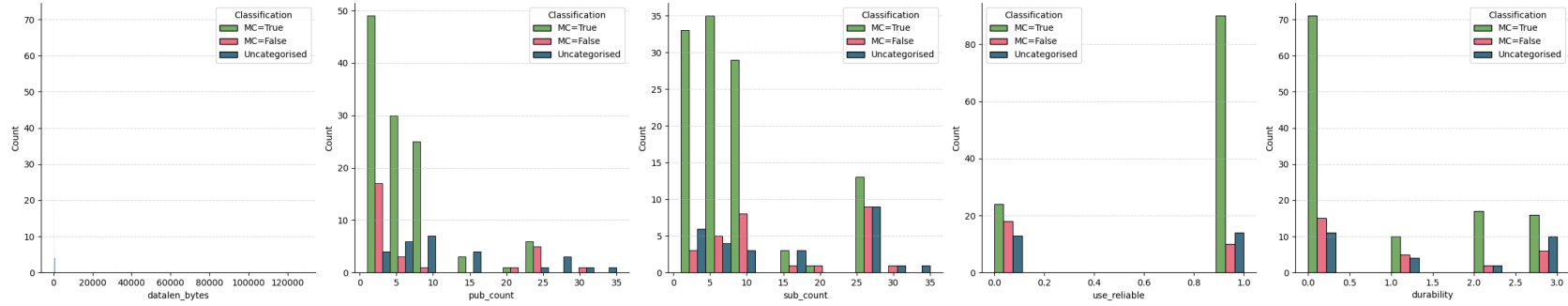


Figure 4.39: Histogram of data distribution for the 95 to 100 percentile range in throughput analysis.

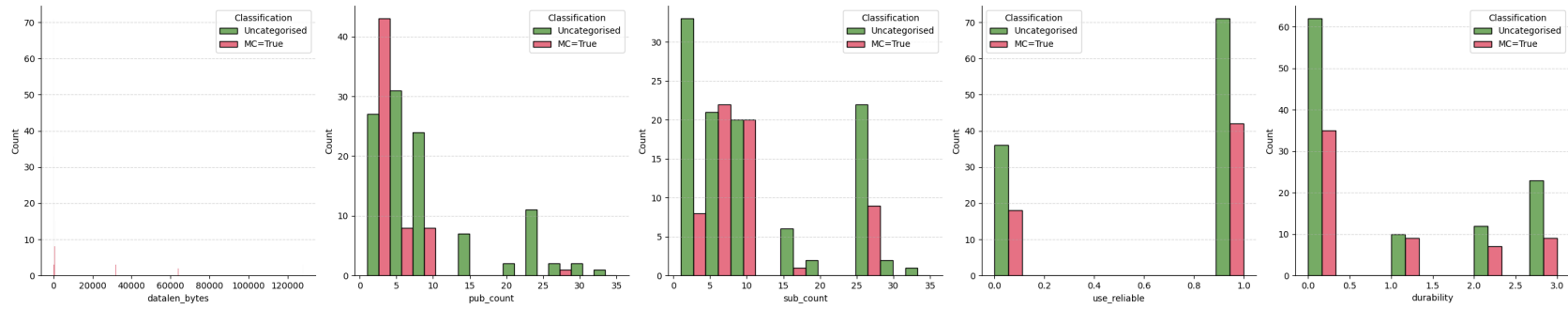


Figure 4.40: Histogram of data distribution for the 0 to 100 percentile range in throughput analysis.

95 to 100 Percentile Figure 4.39 shows the histograms of parameter values for the 95 to 100th percentile. Of the 167 cases in this range, 18 configurations were identified where UC communication consistently outperformed MC, compared to 131 configurations favouring MC, and 18 uncategorised cases. All classifications had a low number of publishers, though uncategorised cases had a low peak at 10 publishers. On top of this, all classifications had a low number of subscribers, with a peak at 25. The classifications differed in the use of reliability where roughly 60 cases were best effort and 40 were reliable for UC. In the case of MC, roughly 90 cases were reliable while around 24 were best effort. Uncategorised cases had an almost equal split between reliable and best effort. These observations indicate that the ratio between reliable and best-effort communication leans more towards reliable in MC cases, while UC leans more towards best-effort.

0 to 100 Percentile The overall distribution is shown in Figure 4.40 for the 0 cases where UC was most performant, 60 cases where MC was most performant, and 107 uncategorised cases. Both cases (MC and uncategorised) had low publisher counts, while subscriber counts ranged from low to medium levels, with a notable peak at 25 (for both publisher and subscriber count) for uncategorised cases. Reliable communication was used in 41 of the MC cases, compared to 18 for best-effort, and all MC experiments were configured with no durability. Meanwhile, reliable communication was more common in uncategorised cases (roughly 71 cases) compared to best-effort (roughly 36 cases), and all cases were configured with a durability of 0. The ratio of reliable to best-effort communication was similar in both classifications, with reliable being more prevalent. The only difference was that MC had a slightly higher ratio of reliable to best-effort communication compared to uncategorised cases. This suggests that reliability may play a role in throughput performance for MC communication and warrants further investigation using statistical analysis. Using all of the insights gained from the individual percentile ranges, a consolidated summary of the key characteristics for each communication type is presented in Table 4.17.

Section	Classification	Cases	Data Length	Publishers	Subscribers	Reliable	Durability
0 to 5	UC	5 of 147	100 B	Low to Mid	Low	Spread	Equal
0 to 5	MC	53 of 147	100 B	Low	Low	Reliable	0
5 to 95	UC	1 of 167	100 B	1	1	Reliable	3
5 to 95	MC	109 of 167	100 B	Low	Low to Mid (spike at 25)	Reliable	0
95 to 100	UC	18 of 167	100 B	Low	Low	Reliable	3
95 to 100	MC	131 of 167	100 B	Low	Low (spike at 25)	Reliable	0
0 to 100	UC	0 of 167					
0 to 100	MC	60 of 167	100 B, 512 B	Low	Low to Mid	Reliable	0

Table 4.17: Summary of case classifications across percentile ranges for throughput analysis.

Statistical Analysis of Throughput

To provide an objective assessment of the impact of various parameters on throughput performance, a statistical analysis was conducted using the χ^2 experiment and the V metric. To reiterate, the χ^2 experiment identifies whether there is a significant association between categorical variables and helps to determine which parameters have a meaningful impact on throughput outcomes where either MC or UC was more performant. The combination of both the χ^2 statistic and the p-value provides a comprehensive understanding of the relationships between the parameters and throughput performance. The statistical significance of these parameters is detailed in Tables 4.19 and 4.18. A p-value below 0.05 indicates a statistically significant association, and the V quantifies the strength of this association.

For the 0 to 5th percentile, the worst-case performance, both publisher count and subscriber count showed a significant relationship with performance, with p-values less than 0.05. The V value for publisher count and subscriber count were 0.40 and 0.45 respectively, indicating a moderate association. In the 5 to 95th percentile, the subscriber count was the only parameter found to be highly significant, with a p-value of 1.3e-07 and a V of 0.5. This relationship becomes even more pronounced in the 95 to 100th percentile, the most-performant percentile range, where the p-value for subscriber count is 1.9e-18 and the V is 0.65, suggesting a strong association. When considering the entire distribution, data length, publisher count, and subscriber count all demonstrated a statistically significant impact on throughput. The p-values were 0.0002, 1.1e-05, and 1.9e-07 respectively. The corresponding V values of 0.46, 0.48, and 0.57 indicate moderate to strong associations.

Throughout all percentile ranges, subscriber count consistently emerged as the most influential parameter with the highest V value of 0.65 in the 95 to 100th percentile. Publisher count also showed a significant impact, particularly in the 0 to 5th percentile with a V of 0.40 and in the overall distribution with a V of 0.48. Data length was only significant in the overall distribution, with a V of 0.46.

Parameter	0 to 5 V	5 to 95 V	95 to 100 V	0 to 100 V
Data Length (B)	0.3107	0.2638	0.1672	0.4589
Publisher Count	0.3985	0.1908	0.2506	0.4838
Subscriber Count	0.4451	0.4829	0.6484	0.5731
Reliability	0.0694	0.1078	0.1810	0.0240
Durability	0.1506	0.1149	0.1251	0.1076

Table 4.18: V statistics for throughput analysis across percentile ranges. Highlighted values correspond to statistically significant parameters (p-value < 0.05).

Parameter	0 to 5 χ^2	0 to 5 P-Value	0 to 5 DoF	5 to 95 χ^2	5 to 95 P-Value	5 to 95 DoF	95 to 100 χ^2	95 to 100 P-Value	95 to 100 DoF	0 to 100 χ^2	0 to 100 P-Value	0 to 100 DoF
Data Length (B)	28.3829	0.1006	20	23.2459	0.2769	20	9.3423	0.9786	20	35.1637	0.0002	10
Publisher Count	46.6792	0.0002	18	12.153	0.8392	18	20.97	0.2809	18	39.0899	1.1e-05	9
Subscriber Count	58.2401	0.0001	24	77.9012	1.3e-07	24	140.442	1.9e-18	24	54.8502	1.9e-07	12
Reliability	0.7079	0.7019	2	1.9408	0.3789	2	5.4713	0.0649	2	0.0966	0.7560	1
Durability	6.6667	0.3528	6	4.4129	0.6210	6	5.2263	0.5151	6	1.9346	0.5861	3

Table 4.19: Chi-square and p-value statistics for throughput analysis across percentile ranges.

4.4 Analysis on the Impact of Data Length

This section presents an analysis of the impact of data length on the performance of the DDS. The size of the data being sent through DDS, also known as the data length, is a parameter that can significantly influence performance in terms of latency and throughput. Larger data lengths can increase network transmission time and processing overhead, potentially leading to higher latency and lower throughput. Conversely, smaller data lengths may result in more performant communication. However, the interaction between small data lengths and reliability could lead to worse performance e.g. low data lengths with best effort communication would lead to overloaded networks and increased packet loss resulting in higher latency and lower throughput. This section aims to investigate the direct impact of data length on DDS latency and throughput and to also explore how this parameter interacts with other configuration parameters such as reliability, MC, publisher count, subscriber count, and durability.

4.4.1 Analysis Methodology

The analysis is based on data from experiments where the data length was systematically varied while other parameters were kept constant which allows for a clear isolation of the impact of data length on performance. Many data lengths were used: 32 B, 64 B, 100 B, 128 B, 256 B, 512 B, 1 KB, 1024 B. However, not all mentioned data lengths were simultaneously compared in the same configuration. We therefore present the results depending on the data length comparisons seen in the configurations:

- 100 B vs 512 B
- 100 B vs 1 KB
- 512 B vs 1 KB
- 100 B vs 512 B vs 1 KB
- 32 B vs 64 B vs 128 B vs 256 B vs 512 B vs 1024 B

The results are presented in terms of latency and throughput, with a focus on how these metrics change with different data lengths. Just like in the analysis of MC in Section 4.3, the PAT tool is used to dynamically categorise the performance of each data length according to stochastic ordering on eCDFs with error margins at a 95% confidence interval. Statistical analysis is performed using the χ^2 and V tests to identify significant relationships between data length and performance metrics. The analysis is broken down into two main sections: latency and throughput. Each section is further broken down into the data length comparisons

listed above which are further split into two subsections; distribution analysis and statistical analysis. Distribution analysis involves examining the distribution of the parameters for the best performing data lengths to identify any patterns or trends. Statistical analysis involves applying the χ^2 and V tests to quantify the strength of the relationships observed in the distribution analysis. All analysis is done on four different percentile ranges: 0 to 5, 5 to 95, 95 to 100, and 0 to 100. This is to capture performance across the entire distribution as well as focusing on best-case, worst-case performance scenarios, and in-between.

4.4.2 Latency

100 B vs 1 KB

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
100 B	19	10	4	3
1 KB	0	0	14	0
Uncategorised	8	17	9	24
Total	27	27	27	27

Table 4.20: Counts of best performing data lengths for latency between 100 B and 1 KB.

Distribution Analysis Table 4.20 shows the counts of best performing data lengths across different percentile ranges for 27 cases where 100 B was compared with 1 KB. Figures 4.41, 4.42 and 4.43 show the distributions of the parameters for the best, normal and worst-case performance ranges, 0 to 5, 5 to 95 and 95 to 100 percentiles respectively.

In the best-case performance (0 to 5th percentile), 100 B outperformed 1 KB in 19 of the 27 configurations while 8 configurations were uncategorised indicating no clear winner between the two data lengths. Of these 19 cases, the majority used between 1 to 10 publishers and subscribers, 17 cases used best-effort compared to 2 cases using reliable communication, 7 cases used UC compared to 12 cases using MC, and the majority of cases had a durability setting of 0 (17 cases). Whilst 100 B had subscriber counts in the lower range (1 to 10), uncategorised cases had a more even distribution across all subscriber counts with a notable peak at 25 subscribers indicating that higher subscriber counts may lead to similar performance between 100 B and 1 KB. This may confirm the hypothesis that involving more subscribers for a given data length (in this case 100 B) can lead to a decrease in latency. However, these results do not explicitly state if the latency of 100 B decreases to the level of 1 KB or if the latency of 1 KB increases to the level of 100 B and so further investigation is required to confirm this. All 8 cases that

were uncategorised used best-effort communication, and 6 of these cases used UC while 2 cases used MC.

These findings suggest that for low-latency applications, particularly those with lower publisher and subscriber counts, a smaller data length of 100 B is generally more advantageous as one would expect. Furthermore, the use of best-effort communication and MC appears to enhance the performance of the smaller data length.

For the worst-case performance percentile range (95th to 100th), the advantage of 100 B over 1 KB is reversed with 4 cases in favour of 100 B and 14 cases in favour of 1 KB, with 9 cases being uncategorised. The distributions that differed between the two data lengths suggest that these parameters have an impact on the performance of the data lengths. These include, the publisher count, where 100 B favoured 10 publishers while 1 KB favoured lower publisher counts (1 and 5), the subscriber count, where 1 KB favoured 20 subscribers while 100 B had a more even distribution, and the MC setting, where 1 KB favoured UC while 100 B had a more even distribution. Both data lengths only used best-effort communication and no durability. However, conclusions cannot be drawn from these findings without further investigation which is done in the statistical analysis proceeding this subsection.

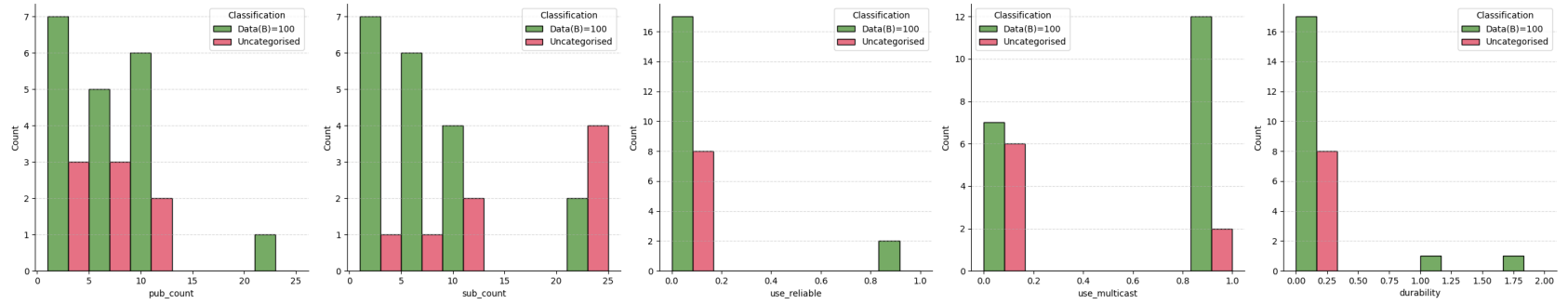


Figure 4.41: Histogram showing distribution of best performing data lengths per configuration parameter between 0 and 5 percentiles for comparison of 100 B and 1 KB.

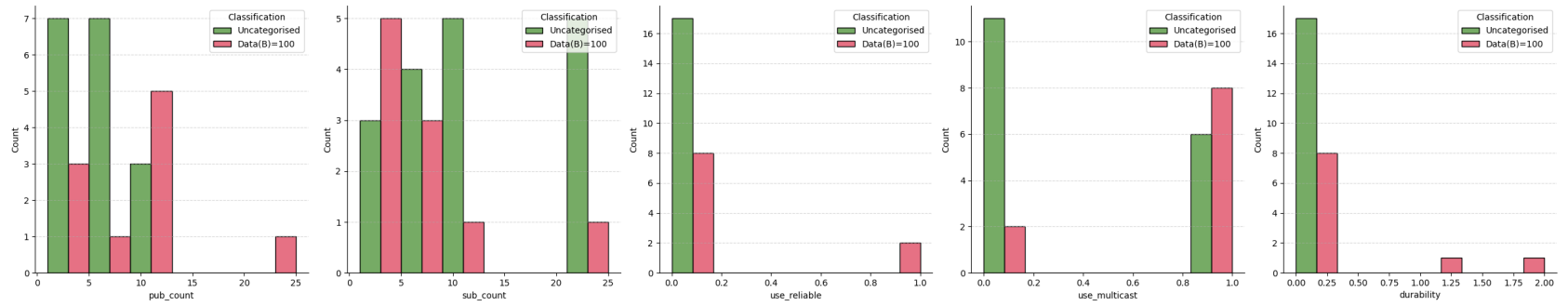


Figure 4.42: Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 100 B and 1 KB.

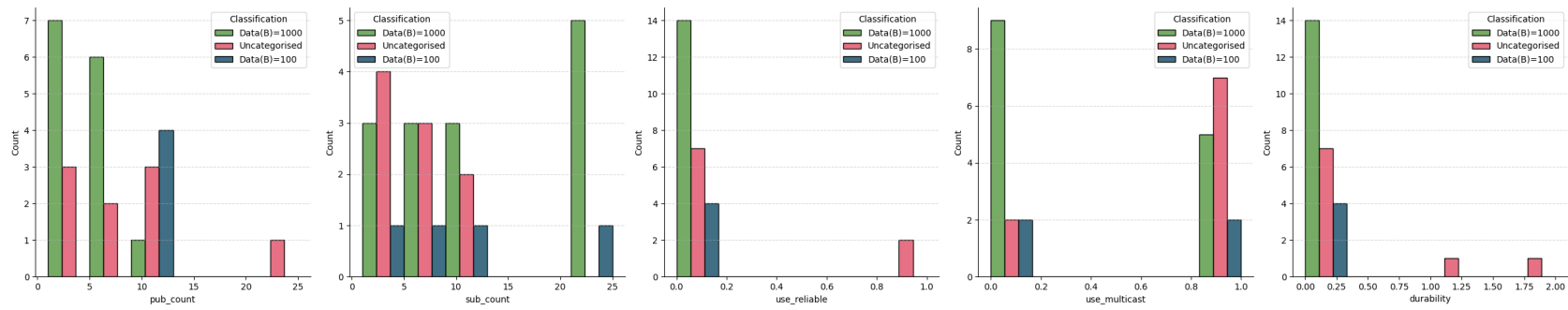


Figure 4.43: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B and 1 KB.

Statistical Analysis The publisher count is the only parameter shown to have statistical significance though specifically between the 95 to 100 percentile range with an χ^2 value of 15.44, a p-value of 0.02 and 6 degrees of freedom. The V value is 0.5348 indicating a moderate relationship between the publisher count and the data length. The distribution analysis in the previous subsection suggested that lower publisher counts (1 and 5) favoured 1 KB while a higher publisher count (10) favoured 100 B which shows the direction of this relationship.

100 B vs 512 B

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
100 B	1	2	7	0
512 B	0	0	3	0
Uncategorised	9	8	0	10
Total	10	10	10	10

Table 4.21: Counts of best performing data lengths for latency between 100 B and 512 B.

Distribution Analysis Table 4.21 shows the counts of best performing data lengths for latency between 100 B and 512 B across different percentile ranges for 10 cases. There were a limited number of configurations (10) and should therefore be treated with caution. Across the entire percentile range, all 10 configurations were uncategorised, indicating no clear winner. In the best-case performance range (0 to 5 percentiles), there was only 1 case where 100 B was most performant while the remaining 9 cases were uncategorised. This trend continues in the 5th to 95th percentile range, where there were 2 cases where the 100 B data length outperformed the 512 B data length before the trend reverses in the 95th to 100th percentile range where there were 7 cases where the 100 B data length outperformed the 512 B data length and 3 cases where the 512 B data length outperformed the 100 B data length. The reversal of the trend in the 95th to 100th percentile range suggests that the 100 B data length may be more advantageous in higher latency scenarios as expected. However, this advantage is not clear in the lower latency scenarios (0 to 5 and 5 to 95 percentiles) where the 100 B data length only outperformed the 512 B data length in 1 and 2 cases respectively of the 10.

An emphasis is placed on the 95th to 100th percentile range as this represents the worst-case performance scenarios which are often critical in real-time applications. Figure 4.44 shows the distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles. Of the 7 cases where the 100 B data length outperformed the 512 B data length, 2 cases had 5 publishers, 2 cases had 10 publishers, and 3 cases had 25 publishers. Meanwhile,

of the 3 cases where the 512 B data length outperformed the 100 B data length, 2 cases had 1 publisher and 1 case had 5 publishers. In terms of subscriber count, the 100 B data length had 4 cases with 5 subscribers, 1 case with 15 subscribers and 2 cases with 25 subscribers. The 512 B data length had 2 cases with 5 subscribers and 1 case with 20 subscribers. The cases where 100 B outperformed 512 B had a ratio of 4 cases using UC and 3 cases using MC. The cases where 512 B outperformed 100 B had a ratio of 2 cases using MC and 1 case using UC, a slight reversal of the trend seen in the cases where 100 B outperformed 512 B. The slight differences in the distributions of the parameters between the two data lengths are seen in the publisher count, where 100 B favoured higher publisher counts (10 and 25) while 512 B favoured lower publisher counts (1 and 5), and the MC setting, where 100 B favoured UC while 512 B favoured MC.

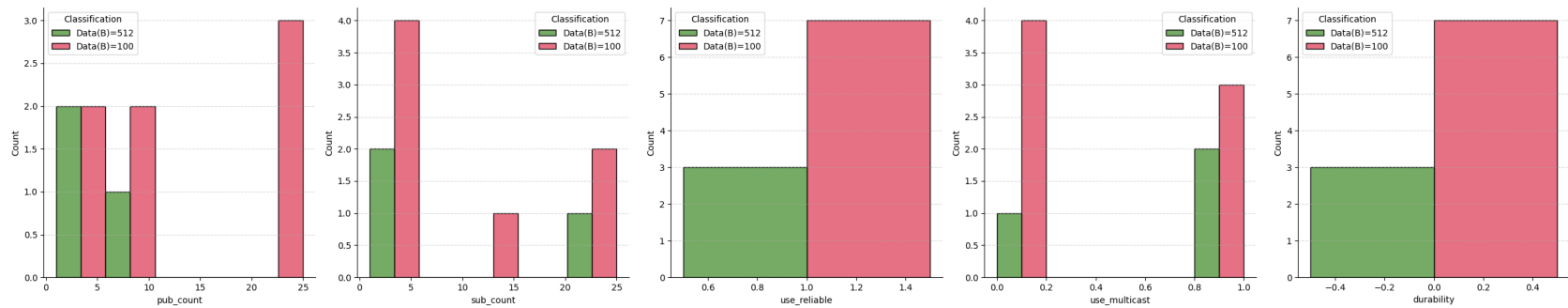


Figure 4.44: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B and 512 B.

512 B vs 1 KB

In the single case comparing 512 B and 1 KB data lengths, the 512 B option demonstrated superior performance across all percentile sections. However, the statistical analysis, including both χ^2 and V tests, did not produce any significant results, with all p-values exceeding 0.05.

100 B vs 512 B vs 1 KB

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
100 B	3	1	6	0
512 B	0	3	9	0
Uncategorised	14	13	2	17
Total	17	17	17	17

Table 4.22: Counts of best performing data lengths for latency between 100 B, 512 B, and 1 KB.

Distribution Analysis The three previous comparisons (100 B vs 1 KB, 100 B vs 512 B, and 512 B vs 1 KB) were pairwise analyses of the data lengths. Here, all three data lengths are compared simultaneously. Table 4.22 shows the counts of best performing data lengths for latency between 100 B, 512 B, and 1 KB across different percentile ranges. 1 KB did not outperform either of the other data lengths in any configuration or percentile range which is consistent with the findings from the previous pairwise comparisons and also with expectations.

Between the 0th and 5th percentiles, the most-performant range, the 100 B data length outperformed both other options in 3 out of 17 cases, while all other cases were uncategorised. This indicates that in low-latency scenarios, the 100 B data length is generally more advantageous, though not overwhelmingly so. Within the 5th to 95th percentile range, the 512 B data length outperformed the 100 B option in 3 out of 17 cases, while the 100 B data length was superior in only 1 case and all other cases were uncategorised. This suggests that while 512 B may not outperform 100 B in low-latency scenarios, it does have a performance advantage in mid-range latency scenarios. In the 95th to 100th percentile range, the 512 B data length outperformed the 100 B option in 9 out of 17 cases, while the 100 B data length was superior in the remaining 6 cases. This indicates that in high-latency scenarios, the 512 B data length is generally more advantageous which is counter intuitive as one would expect the smaller data length (100 B) to be more performant in all scenarios, especially those with higher latency. Finally, looking at the entire distribution (0 to 100 percentiles), all 17 cases were uncategorised indicating no clear winner between all data lengths.

A focus is placed on the 95th to 100th percentile range as this represents the worst-case performance scenarios and is the range with the least number of uncategorised cases (2) indicating a clearer distinction between the data lengths. Figure 4.45 shows the distribution of the parameter values. The distributions show that 512 B favoured 1 publisher (6 cases) while 100 B favoured 5 and 10 publishers equally (3 cases each). In terms of subscriber count, 512 B favoured 5 subscribers (4 cases) while 100 B favoured 10 subscribers (3 cases) though both data lengths had a more even distribution across the other subscriber counts. Both data lengths only used reliable communication and no durability and both had a similar distribution between UC and MC with 512 B having a more balanced distribution (5 cases for UC and 4 cases for MC) while 100 B had a pronounced preference for UC (5 cases for UC and 1 cases for MC). The differences in the distributions for the publisher count, subscriber count, and MC setting suggest that these parameters have an impact on the performance of the data lengths. Further analysis using statistical methods is used to confirm these findings in the following subsection.

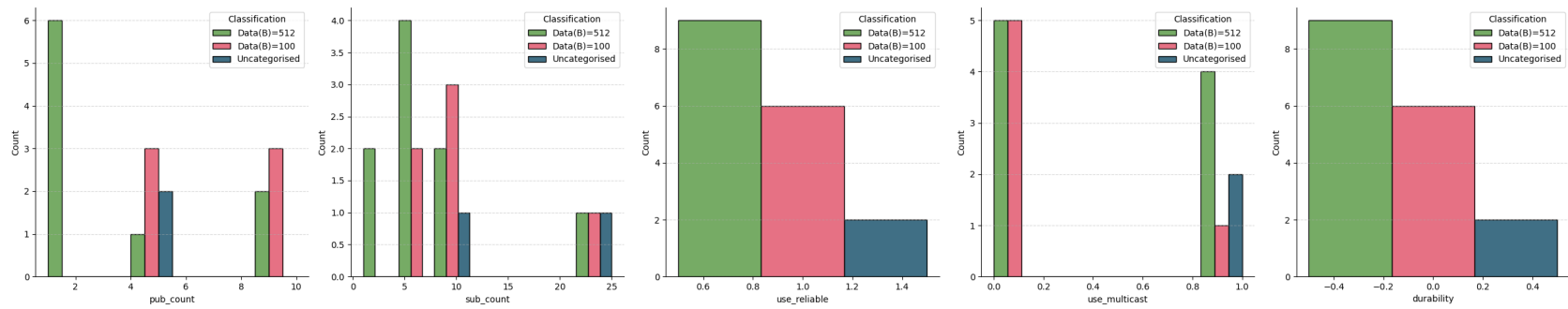


Figure 4.45: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B, 512 B, and 1 KB.

Statistical Analysis Across all χ^2 tests for all percentile ranges, only the publisher count parameter demonstrated statistical significance, specifically in the 95th to 100th percentile range with a p-value of 0.02, χ^2 value of 11.18, 4 DoF, and a V value of 0.57 indicating a strong relationship between publisher count and data length performance in high-latency scenarios. Based off the distribution analysis, it appears that lower publisher counts (1) favour the 512 B data length while higher publisher counts (5 and 10) favour the 100 B data length demonstrating the direction of this relationship.

32 B vs 64 B vs 128 B vs 256 B vs 512 B vs 1024 B

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
64 B	0	0	1	0
128 B	0	0	2	0
512 B	0	0	1	0
Uncategorised	9	9	5	9
Total	9	9	9	9

Table 4.23: Counts of best performing data lengths for latency between 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B. Only data lengths with a least 1 case of best performance are shown.

Distribution Analysis A total of 9 configurations compared the 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B data lengths. Table 4.23 shows the counts of best performing data lengths. Data lengths that are not seen in the table but mentioned otherwise were not the best performing option in any configuration or percentile range. As anticipated, the performance of the different data lengths intersected across the entire percentile range, leading to no definitive optimal choice. In fact, all 9 configurations were uncategorised across the ranges of 0 to 5 percentiles, 5 to 95 percentiles, and 0 to 100 percentiles. The only section where a definitive optimal data length emerged was the 95th to 100th percentile range, representing the worst-case performance scenarios. Figure 4.46 shows the distribution of best performing data lengths per configuration parameter within this percentile range and highlights the variability in parameter distributions. These results should be interpreted with caution due to the limited number of configurations (9) that compared these data lengths. Nonetheless, the results are provided for completeness and to offer insights into potential trends.

The single case where 64 B outperformed all other data lengths (especially 32 B) used 1 publisher, 3 subscribers, best-effort communication, MC, and no durability. This results is somewhat counter intuitive as one would expect the smallest data length (32 B) to be optimal

in this scenario. The 2 cases where 128 B outperformed all other data lengths used 1 publisher, best-effort communication, MC, and no durability but had 6 and 9 subscribers respectively. The single case where 512 B outperformed all other data lengths used 1 publisher, 3 subscribers, reliable communication, UC, and no durability. All cases used 1 publisher and no durability, indicating that these parameters may not have a significant impact on the performance of the data lengths in this comparison. The parameters that differed between the cases were the subscriber count, reliability setting, and MC setting. Both 64 B and 512 B used 3 subscribers while 128 B used 6 and 9 subscribers respectively. All cases used best-effort communication except for the 512 B case which used reliable communication and 512 B used UC while all other cases used MC. These findings suggest that the subscriber count, reliability setting, and MC setting may have an impact on the performance of the data lengths in this comparison.

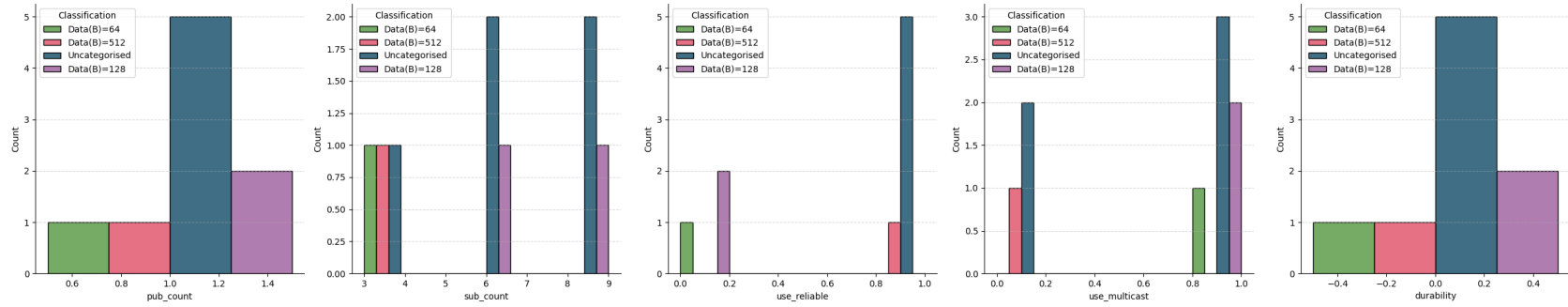


Figure 4.46: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles.

Statistical Analysis Of all sections, and all parameters, only one statistically significant relationship was found. For the 95th to 100th percentile, the reliability setting was found to have a statistically significant relationship with the optimal data length, with a p-value of 0.03, a χ^2 value of 9, 3 DoF, and a V value of 1. This indicates a perfect association between the reliability setting and worst-case latency performance in this comparison. The distribution analysis showed that the 512 B data length, which was optimal in one case, used reliable communication while the other data lengths used best-effort communication indicating that reliable communication may favour larger data lengths in high-latency scenarios. However, as previously mentioned, these results should be interpreted with caution due to the limited number of configurations (9) that compared these data lengths.

Discussion of Latency Analysis

The analysis was broken down into distribution analysis and statistical analysis for all data length comparisons. Table 4.24 summarises all statistically significant cases across all data lengths and percentile ranges. The worst-performing percentile range (95 to 100) is seen as the most impactful in 3 of the 4 cases. Publisher count and reliability were identified as the parameters that statistically significantly impact the outcome of the most performant data length.

Parameter	Data Lengths Compared	Percentile Range	χ^2	p-value	DoF	V
Publisher Count	100B, 512B, 1KB	95 to 100	11.18	0.02	4	0.57
	100B, 1KB	95 to 100	15.44	0.02	6	0.53
		0 to 100	8.02	0.05	3	0.54
Reliability	32B, 64B, 128B, 256B, 512B, 1024B	95 to 100	9.00	0.03	3	1.00

Table 4.24: Summary of all statistically significant relationships between parameter and data lengths for latency performance.

In the case comparing 100 B to 1 KB, 100 B was the most performant with 10 publishers, while 1 KB was the most performant with 1 or 5 publishers (there was 1 case with 10 publishers). This demonstrates a preference of less publishers for larger data lengths (1 KB in this case) and a preference of more publishers for smaller data lengths (100 B in this case). Therefore, this identifies an inverse relationship where smaller data lengths outperform larger data lengths in terms of worst-case latency with more publishers and where larger data lengths outperform smaller data lengths in terms of worst-case latency with less publishers. This relationship is further confirmed when simultaneously comparing 100 B to 512 B and 1 KB. 100 B was seen

to favour more publishers where half of the cases involved 5 publishers while the other half involved 10. Meanwhile, 512 B favoured less publishers where all cases used 1 publisher. 1 KB did not outperform all other data lengths (100 B and 512 B) for any of these cases in the worst-performing range. This confirms this inverse relationship between data length and publisher count and provides insight into the interaction between various parameters on the impact of DDS performance.

Reliability was seen as a statistically significantly impactful parameter on the performance of varied data lengths in a single case; when comparing 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B in the worst-case percentile range (95 to 100). In this range, only 4 cases identified the most performant data length. The single case where 64 B outperformed all other data lengths (including 32 B) used best effort communication, 128 B outperformed all others for 2 cases and also used best effort communication, and the largest data length to outperform all others was 512 B which used reliable communication. This switch identifies a relationship that is further supported by the statistical tests. These results demonstrate that smaller data lengths perform better when using best-effort communication as seen with 64 B and 128 B and larger data lengths perform better with reliable communication as exemplified by 512 B.

Summary of Latency Analysis

The analysis of latency across various data lengths reveals a complex and non-linear relationship, influenced by multiple configuration parameters. Generally, for best-case performance, characterised by the lower latency percentiles (0-5th), smaller data lengths such as 100 B demonstrated a consistent advantage. This aligns with the expectation that smaller payloads incur less transmission and processing overhead. However, this trend did not hold uniformly across all scenarios, particularly in the worst-case performance percentiles (95-100th). Counter-intuitively, larger data lengths like 512 B and 1 KB were often found to be more performant in these high-latency situations. Statistical analysis identified the publisher count as a consistently significant factor influencing this relationship. Specifically, in high-latency scenarios, lower publisher counts tended to favour larger data lengths, whereas higher publisher counts showed an advantage for smaller data lengths. Other parameters, including the reliability setting, also showed statistical significance in certain comparisons, suggesting that the choice of communication protocol can modulate the impact of data length on latency. For instance, reliable communication appeared to favour larger data lengths in high-latency conditions when comparing a wide range of data lengths. It is crucial to note that many comparisons were

based on a limited number of configurations, thus the findings should be interpreted with caution. Ultimately, the selection of an optimal data length for latency is not straightforward and requires careful consideration of the entire operational context, including publisher and subscriber counts, reliability requirements, and the specific latency percentile of interest to the application.

4.4.3 Throughput

100 B vs 512 B

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
512 B	1	5	9	1
Uncategorised	9	5	1	9
Total	10	10	10	10

Table 4.25: Counts of best performing data lengths for throughput between 100 B and 512 B. 100 B did not outperform 512 B in any case.

Distribution Analysis Table 4.25 summarises all cases of best performing data length between 100 B and 512 B. The results show that 100 B never outperforms 512 B in terms of throughput for all percentile ranges which conforms with expectations. However, this does not mean that the performance of 100 B is always worse than 512 B. On the contrary, in the worst-case performance range (0 to 5th percentile), 9 of the 10 cases are uncategorised indicating crossover points between the 2 eCDFs. Within the middle range (5 to 95th percentile), there are an equal number of cases (5) where 512 B outperforms 100 B and where the winner is uncategorised. Finally, in the best-performing range (95 to 100th percentile), a mirror effect is observed of the worst-performing range. In the worst case range, 512 B outperformed 100 B for 1 case while the other 9 were uncategorised and the opposite is seen in the best case range where 512 B outperforms 100 B for 9 cases with the final case being uncategorised. This indicates that, in 9 of 10 cases, 512 B will produce higher throughput when both data lengths are at peak performance. Meanwhile, looking at the entire distribution as a whole, the results repeat what was seen in the first 5 percentiles, 9 of the 10 are uncategorised.

Statistical Analysis Results of the χ^2 statistical test only produced one parameter with a p-value < 0.05 . The subscriber count had a p-value of 0.046 between the 5th and 95th percentile with an χ^2 of 8, DoF of 3, and a V of 0.89 which indicates a significantly strong relationship

between the number of subscribers used and whether 512 B outperforms 100 B or vice versa. However, these results should be interpreted with caution given the low number of samples (10 cases) and the border-line p-value (0.046).

100 B vs 1 KB

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
100 B	1	3	4	2
1 KB	1	6	21	1
Uncategorised	15	18	2	24
Total	17	27	27	27

Table 4.26: Counts of best performing data lengths for throughput between 100 B and 1 KB.

Distribution Analysis Table 4.26 shows the counts of best performing data lengths for throughput between 100 B and 1 KB. The inconsistency in the number of cases across percentile ranges is attributed to the lack of samples in certain configurations, leading to their exclusion from specific percentile calculations.

Figures 4.47, 4.48, 4.49, and 4.50 show the distribution of best performing data lengths per configuration parameter within each percentile range.

In the 0 to 5th percentile range, the 100 B data length was found to be optimal in one specific configuration, which consisted of 25 publishers, 5 subscribers, reliable UC communication, and a durability of 0. The 1 KB data length was optimal in a different configuration, which included 1 publisher, 3 subscribers, best-effort MC communication, and a durability of 1. Theoretically, one would expect the 1 KB data length to perform better in this scenario due to its larger size, which should allow for more efficient use of network resources and this does hold true in 1 of the cases. However, a majority of cases (15) were uncategorised, indicating that the performance of the two data lengths intersected in most configurations. This is a surprising result, as these findings suggest that the 100 B data length can be competitive with the 1 KB data length in certain scenarios, even in the worst-case performance range.

Within the 5th to 95th percentile, the 100 B data length was advantageous in 3 configurations with low publisher and subscriber counts, reliable communication, and a preference for UC over MC (2 to 1 ratio), all with no durability. Meanwhile, the 1 KB data length excelled in 6 configurations with 1 to 5 publishers, 1 (2 cases), 10 (1 case), and 25 (3 cases) subscriber counts, a 2 to 1 ratio in favour of best-effort communication, and a preference for MC over UC (5 to 1 ratio), with a mix of durability settings (0, 1 and 2). In the best-performing range, the

95th to 100th percentile, where 100 B outperformed 1 KB in 4 configurations, the 100 B data length showed better performance with low publisher counts, low to medium subscriber counts, best-effort reliability, and a preference for MC over UC (3 to 1 ratio), all with a durability of 0. For this percentile range, the 1 KB data length outperformed 100 B in 21 configurations. This large number of configurations conforms with expectations, as the larger data length should theoretically perform better in best-performance scenarios. Of these 21 configurations, 1 to 10 publishers were used with 5 cases having 1 publisher and 8 cases having 5 and 10 publishers each. Subscriber counts varied from 1 to 25, with peaks at 1 (6 cases), 10 (5 cases), and 25 (6 cases). A preference for best-effort communication was observed (19 to 2 ratio), with a slight preference for MC over UC (11 to 10 ratio). Durability settings were set to 0 in all but 2 cases, which used durability settings of 1 and 2 respectively. The differing parameter settings for this specific percentile range are seen for the publisher and subscriber count where 100 B favoured lower counts while 1 KB had a wider distribution. Best-effort communication was favoured by both data lengths, though 100 B had no cases using reliable communication while 1 KB had 2 cases.

Across the entire percentile range, the 100 B data length was superior in 2 cases, both involving 1 publisher, 1 subscriber, best-effort communication, and a durability of 0. The 2 cases differed in their communication protocol, with one using UC and the other MC. The 1 KB data length was optimal in one case, which included 5 publishers, 25 subscribers, best-effort UC communication, and a durability of 0. All other cases (24) were uncategorised, indicating intersecting performance between the two data lengths in most configurations.

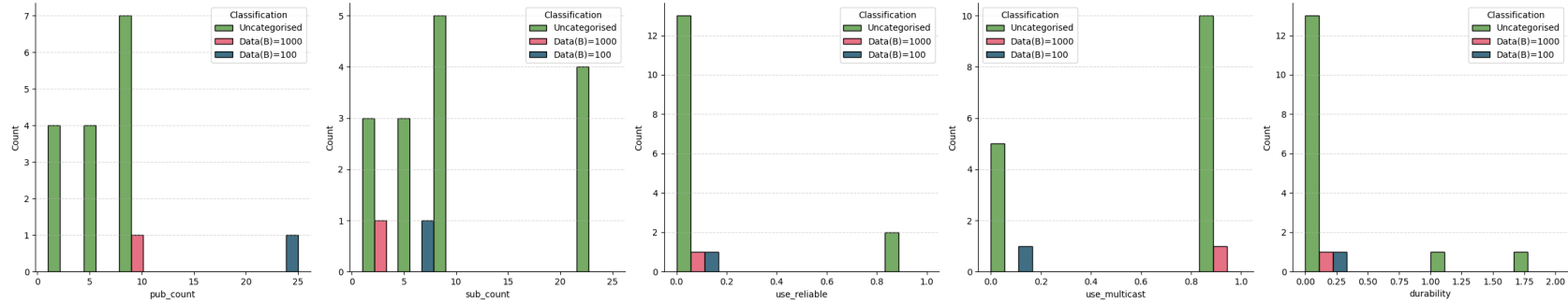


Figure 4.47: Histogram showing distribution of best performing data lengths per configuration parameter between 0 and 5 percentiles for comparison of 100 B and 1000 B.

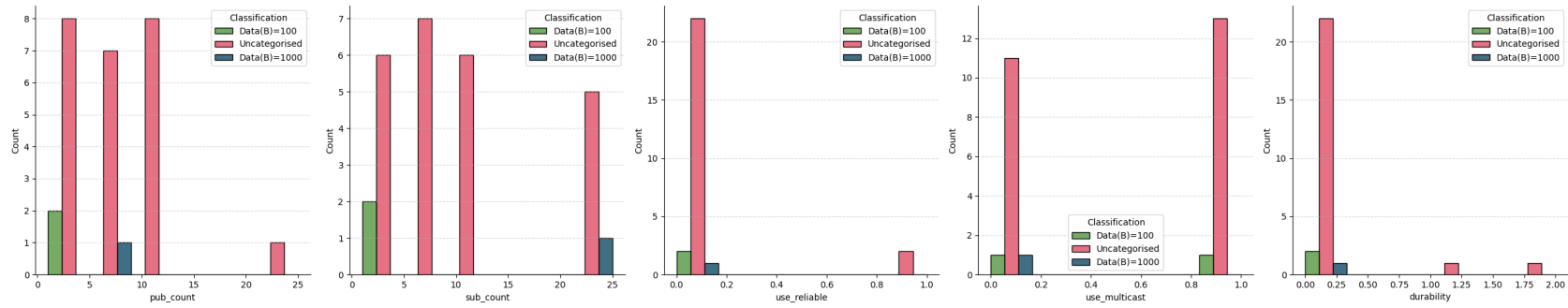


Figure 4.48: Histogram showing distribution of best performing data lengths per configuration parameter between 0 and 100 percentiles for comparison of 100 B and 1000 B.

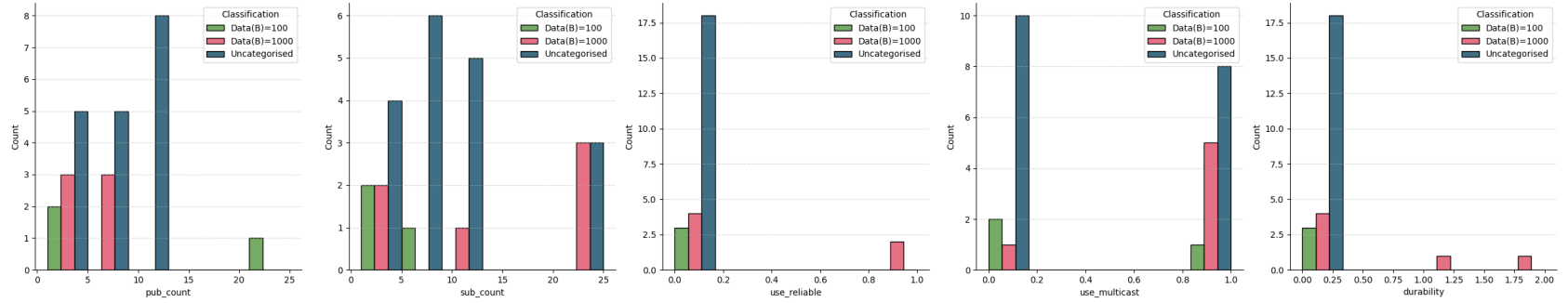


Figure 4.49: Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 100 B and 1000 B.

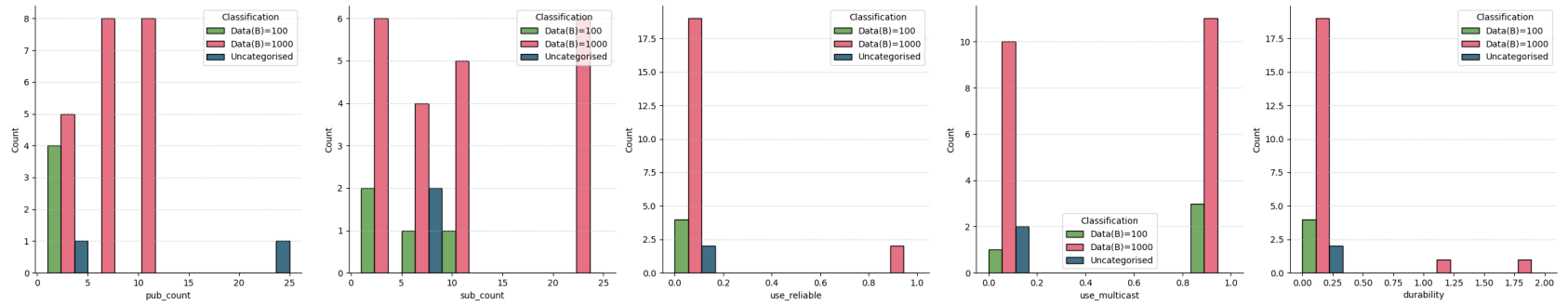


Figure 4.50: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B and 1000 B.

Parameter	0 to 5 Cramers V	5 to 95 Cramers V	95 to 100 Cramers V	0 to 100 Cramers V
Publisher Count	0.73	0.53	0.64	0.33
Reliability	0.13	0.53	0.15	0.10

Table 4.27: V correlation analysis for throughput between 100 B and 1 KB. Highlighted values indicate statistical significance (p-value < 0.05).

Statistical Analysis Table 4.27 presents the results of the V correlation analysis for throughput between 100 B and 1 KB and Tables 4.28 and 4.29 shows the corresponding χ^2 values. Of the 5 parameters, across the 4 different percentile ranges, only 4 cases had a p-value < 0.05 across 2 parameters; publisher count and reliability. Reliability between the 5th and 95th percentiles had a p-value of 0.02, an χ^2 of 7.56 and a DoF of 2 and the publisher count had multiple p-values < 0.05 observed in the range of 0 to 5, 5 to 95, and 95 to 100 with values of 0.01, 0.02, and 0.006 respectively. This suggests localised imbalances or pronounced effects within these ranges, potentially arising from clustering of extreme values in the tails (0-5 and 95-100) or variability in the central bulk (5-95) that violates assumptions of uniformity. Conversely, the analysis across the entire distribution (0 to 100) did not reveal any statistically significant relationships, with p-values exceeding the 0.05 threshold for all parameters. This discrepancy may occur because opposing deviations in the percentile ranges effectively counterbalance one another when amalgamated; excesses in one range might offset deficits in another. Such outcomes emphasise the need to analyse the ranges as well as the entire distribution to uncover such hidden findings.

Parameter	0 to 5 χ^2	0 to 5 P- Value	0 to 5 DoF	5 to 95 χ^2	5 to 95 P- Value	5 to 95 DoF
Publisher Count	18.13	0.01	6	15.15	0.02	6
Reliability	0.30	0.86	2	7.56	0.02	2

Table 4.28: χ^2 correlation analysis (0–95 percentiles). Highlighted values indicate statistical significance ($p < 0.05$).

Parameter	95 to 100 χ^2	95 to 100 P- Value	95 to 100 DoF	0 to 100 χ^2	0 to 100 P- Value	0 to 100 DoF
Publisher Count	22.44	0.00	6	5.99	0.42	6
Reliability	0.62	0.73	2	0.27	0.87	2

Table 4.29: χ^2 correlation analysis (95–100 and overall). Highlighted values indicate statistical significance ($p < 0.05$).

100 B vs 512 B vs 1 KB

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
512 B	1	10	11	1
1 KB	0	1	4	0
Uncategorised	16	6	2	16
Total	17	17	17	17

Table 4.30: Counts of best performing data lengths for throughput between 100 B, 512 B, and 1 KB. 100 B did not outperform the other data lengths in any case.

Distribution Analysis A total of 17 cases compared the throughput performance of 100 B, 512 B, and 1 KB simultaneously. The counts of the most performant data lengths are seen in Table 4.30. In all percentile ranges (including overall), 100 B never outperformed the 2 other data lengths. Within the worst-case range (0 to 5) and the overall range (0 to 100), 512 B outperformed all others for 1 of the 17 cases with the other 16 being uncategorised. This single case involved 1 publisher, 1 subscriber, reliable communication using UC with no durability. All of these individual parameter values were also observed in the uncategorised cases making it hard to draw any insights. The fact that 512 B outperforms 1 KB in terms of worst-case performance is of interest as one would expect the opposite to happen, even during worst-case performance. Further investigation into this single experiment may provide insights into why this specific collection of parameter values results in better performance for 512 B over 1 KB.

Between the 5th and 95th percentile, 1 KB outperformed all others in 1 case while 512 B outperformed all others in 10 cases. The remaining 6 cases were uncategorised. This ratio favouring 512 B as the most performant is continued in the best-performing range (95 to 100) where 4 cases indicate 1 KB outperformed all other data lengths versus 11 cases demonstrating 512 B as the most performant. This further emphasises the observation that 512 B outperforms 1 KB under the best conditions which is counter-intuitive. An analysis of the distribution of parameter values may provide insight into the types of conditions that are likely to result in similar findings and are shown in Figure 4.51. In the middle range (5 to 95), the single case where 1 KB outperformed 512 B used 5 publishers, 25 subscribers, reliable communication with UC, and no durability. The cases where 512 B outperformed 1 KB (and also 100 B) had a higher concentration with lower numbers of publishers (compared to 5), more cases with lower subscriber counts (compared to 25), all used reliable communication, had an equal number of UC cases compared to MC and all had no durability.

Are these values consistent for the best performing range for cases where 512 B outperformed

1 KB between the 95th and 100th percentile? Figure 4.52 shows the distributions of parameters values within this range and indeed they are similar to what was seen between the 5th and 95th percentile. This consistency provides insights into the conditions where 512 B outperforms 1 KB.

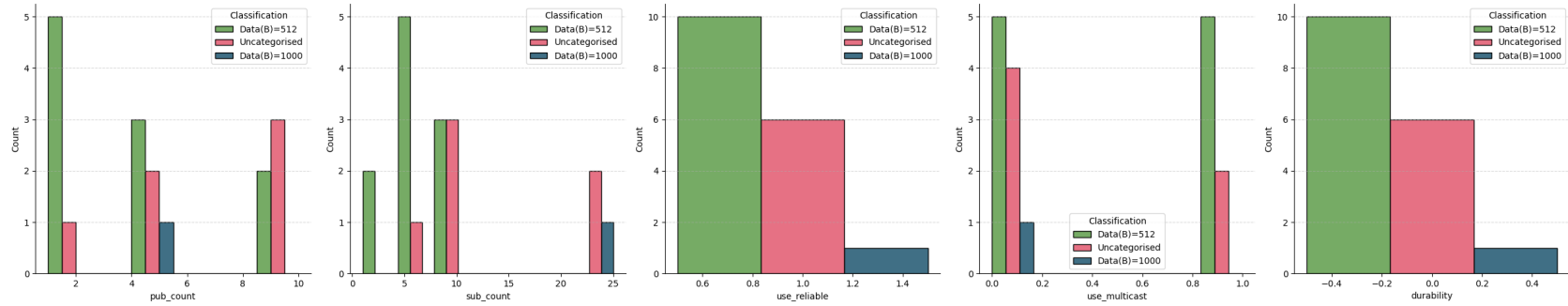


Figure 4.51: Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 100 B, 512 B, and 1 KB.

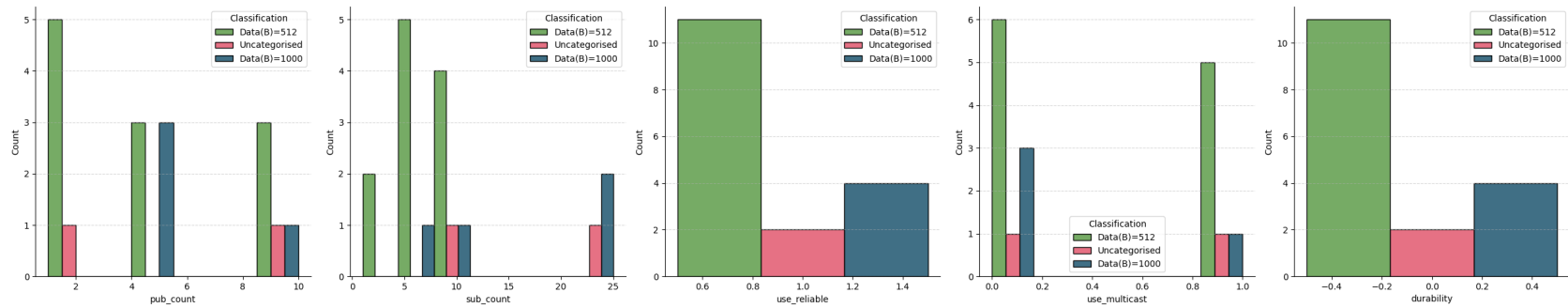


Figure 4.52: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 100 B, 512 B, and 1 KB.

Parameter	0 to 5 Cramers V	5 to 95 Cramers V	95 to 100 Cramers V	0 to 100 Cramers V
Subscriber Count	0.68	0.55	0.48	0.68

Table 4.31: V correlation analysis for throughput between 100 B, 512 B, and 1 KB. Highlighted values indicate statistical significance (p-value < 0.05).

Statistical Analysis Despite the consistency in the parameter value distributions for cases where 512 B outperformed all other data lengths between the 5th to 95th and the 95th to 100th percentiles, the χ^2 results seen in Tables 4.32 and 4.33 only show the subscriber count to have a statistically significant relationship to the data length. The subscriber count had a p-value of 0.0046 in both the first and last 5 percentiles. The V values shown in Table 4.31 demonstrate that the subscriber count has a moderately strong relationship in the first 5 percentiles (with a value of 0.68) compared to the last 5 percentiles (0.48).

Parameter	0 to 5 χ^2	0 to 5 P- Value	0 to 5 DoF	5 to 95 χ^2	5 to 95 P- Value	5 to 95 DoF
Subscriber Count	7.97	0.05	3	10.20	0.12	6

Table 4.32: χ^2 correlation analysis (0–95 percentiles) for throughput between 100 B, 512 B, and 1 KB.

Parameter	95 to 100 χ^2	95 to 100 P- Value	95 to 100 DoF	0 to 100 χ^2	0 to 100 P- Value	0 to 100 DoF
Subscriber Count	7.98	0.24	6	7.97	0.05	3

Table 4.33: χ^2 correlation analysis (95–100 and overall) for throughput between 100 B, 512 B, and 1 KB.

32 B vs 64 B vs 128 B vs 256 B vs 512 B vs 1024 B

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
1024 B	0	4	8	0
512 B	2	0	0	0
Uncategorised	7	5	1	9
Total	9	9	9	9

Table 4.34: Counts of best performing data lengths for throughput between 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B. Only data lengths that outperformed all others are shown.

This comparison simultaneously compares 32 B, 64 B, 128 B, 256 B, 512 B, and 1024 B in terms of throughput. Table 4.34 displays the number of cases for each data length if

it outperformed all others. It is expected that the largest data length (1024 B) outperforms all others, especially so between the 95th and 100th percentiles, the range with the highest throughput values recorded. In the 0 to 5th percentile range, the 512 B data length was optimal in 2 configurations of 1 publisher, 3 and 6 subscribers, best-effort reliability, MC communication, and no durability while the other 7 cases were uncategorised. This suggests that during periods where the least throughput is recorded, there are 2 of 9 cases where 512 B continues to produce the highest throughput, especially when compared to 1024 B. 1024 B never outperforms all other data lengths for any of the 9 cases during this period.

However, there are 4 cases where 1024 B outperforms all others between the 5th and 95th percentiles with the other 5 cases being uncategorised. This is emphasised further in the best performing range where 1024 B outperforms all others in 8 of the 9 cases whilst the last case is uncategorised. This conforms with the expectation that the largest data length is expected to produce the greatest throughput. When observing the distribution as a whole, none of the data lengths ever outperform all others.

Statistical analysis of these results produced no insights as all parameters across all percentile ranges produced p-values greater than 0.05. This is likely attributed to the low number of cases.

Discussion of Throughput Analysis

Parameter	Data Lengths Compared	Percentile Range	χ^2	p-value	DoF	V
Subscriber Count	100B, 512B	5 to 95	8	0.046	3	0.89
	100B, 512B, 1KB	0 to 5	7.97	0.046	3	0.68
		0 to 100	7.97	0.046	3	0.68
Publisher Count	100B, 1KB	5 to 95	15.15	0.02	6	0.53
		95 to 100	22.43	0.001	6	0.64
Reliability		5 to 95	7.56	0.02	2	0.53

Table 4.35: Summary of statistically significant relationships between parameters and data lengths for throughput performance.

A summary of all statistically significant relationships between the most performant data length according to throughput performance and the configuration parameters is shown in Table 4.35. The same parameters seen in the latency summary (publisher count and reliability) are also seen for throughput with the addition of subscriber count. However, the p-values for the subscriber count cases seen round up to 0.05 which subtracts from their statistical significance. Nonetheless, the results are presented to provide probable insight into the complex relationships between the various parameters and their impact on DDS performance.

The publisher count was identified as a statistically significantly impactful parameter on the outcome of data length performance specifically when comparing 100 B to 1 KB. In the 3 of the 27 cases where 100 B outperformed 1 KB between the 5th and 95th percentiles, majority of the cases used 1 publisher while in the 6 of the 27 cases where 1 KB outperformed 100 B, the publisher count was equally split between 5 and 10 publishers. This indicates that smaller data lengths (100 B in this case) perform better with less publishers (1 publisher in this case) and larger data lengths (1 KB in this case) perform better with more publishers (5 or 10 in this case). This suggests a direct relationship between data length and publisher count. This is further supported in the best-case performance range (95 to 100) where the 4 of the 27 cases where 100 B outperformed 1 KB only used 1 publisher. Meanwhile, the majority of the 21 of the 27 cases where 1 KB outperformed 100 B in this high throughput range used 5 or 10 publishers. This strengthens the finding that less data with less publishers and more data with more publishers results in higher throughput.

Reliability was identified as a statistically significantly impactful parameter when comparing 100 B and 1 KB in the 5 to 95 percentile range with a p-value of 0.02. A V value of 0.53 indicates a relationship with moderate strength. In the 3 cases of the 27 where 100 B outperformed 1 KB, all cases used best effort communication. 4 of the 6 cases where 1 KB outperformed 100 B also used best effort communication while the other 2 used reliable communication. This results suggest that both data lengths prefer best effort communication though reliable communication was never seen in the cases where the smaller data length (100 B) outperformed the larger data length (1 KB). This could indicate that larger data lengths in combination with reliable communication are likely to result in better performance while smaller data lengths combined with reliable communication are likely to result in worse performance. Further investigation is necessary to clarify the direction of the relationship between reliability and the data length in terms of throughput performance.

Summary of Throughput Analysis

The investigation into the impact of data length on throughput largely confirms the expected trend that larger data payloads facilitate higher throughput. This was most evident in the best-case performance range (95-100th percentile), where larger data lengths like 1024 B and 1 KB consistently outperformed smaller ones. However, the relationship is not strictly monotonic, and several counter-intuitive results emerged. Most notably, the 512 B data length frequently demonstrated superior throughput compared to the larger 1 KB data length, particularly in

multi-way comparisons. This suggests the existence of an optimal payload size that maximises throughput under certain conditions, beyond which performance may not necessarily improve. Furthermore, in worst-case throughput scenarios (0-5th percentile), it was observed that a mid-range data length (512 B) could outperform a much larger one (1024 B), indicating that the efficiency benefits of larger payloads can diminish during periods of poor performance. Statistical analysis revealed that publisher and subscriber counts are significant factors influencing these outcomes. The number of publishers was a key determinant when comparing 100 B and 1 KB, while the subscriber count was significant when comparing 100 B, 512 B, and 1 KB. The reliability setting also emerged as a significant parameter in some comparisons. The prevalence of uncategorised cases across many percentile ranges highlights that performance often intersects, meaning no single data length is universally optimal. In conclusion, while selecting a larger data length is a reasonable strategy for achieving high throughput, a mid-range payload such as 512 B can offer a competitive, and sometimes superior, performance, depending on the specific system configuration and operating conditions.

4.4.4 Summary

The analysis of data length reveals distinct and sometimes counter-intuitive relationships with key system parameters for both latency and throughput performance. For latency, an inverse relationship with publisher count was identified, wherein smaller data lengths demonstrated superior performance with a higher number of publishers. Conversely, larger data lengths exhibited better latency performance when paired with a lower number of publishers. Furthermore, communication reliability emerged as a significant factor; smaller data lengths favoured best-effort communication, whereas larger data lengths achieved better worst-case latency with reliable communication. For throughput, a direct relationship with publisher count was observed. Smaller data lengths achieved higher throughput with fewer publishers, while larger data lengths benefited from a greater number of publishers. The influence of reliability on throughput indicated a preference for best-effort communication overall. However, the findings suggest that larger data lengths may perform better with reliable communication, although this requires further investigation. These results highlight the complex interplay between data length, publisher count, and reliability, providing critical insights for performance optimisation of DDS systems.

4.5 Analysis on the Impact of Noise in the Network

In real-world deployments, DDS systems often have to operate in imperfect network conditions. This section presents an analysis of the impact of network noise on the performance of the DDS. Network noise, in the form of background traffic and network congestion, can have a significant impact on the reliability and timeliness of data delivery in DDS. This analysis aims to quantify this impact and to evaluate the effectiveness of DDS's reliability mechanisms in mitigating the effects of network noise.

4.5.1 3 Raspberry Pi Analysis

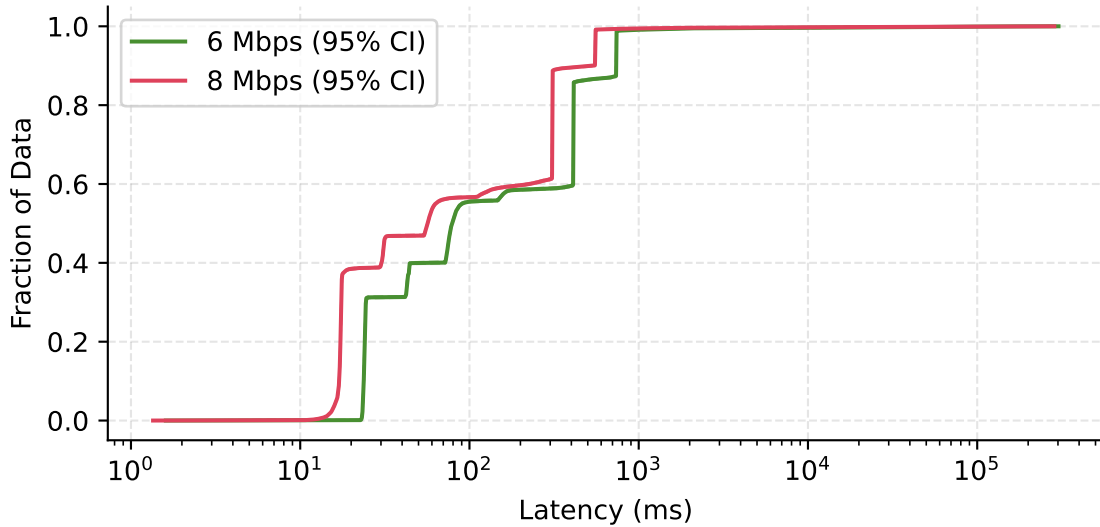


Figure 4.53: eCDFs with 95% confidence intervals of latency (ms) for 6 Mbps (green) versus 8 Mbps (red) bandwidth throttle on 3Pi testbed. x-axis is log-scaled.

Figure 4.53 shows the eCDF of latency for two different bandwidth throttles (6 Mbps as detailed in Subsection 3.4.4 and 8 Mbps as detailed in Subsection 3.4.4) on the 3Pi testbed. The two lines exhibit a clear separation, indicating stochastic ordering. Specifically, a higher bandwidth throttle leads to increased latency, thereby demonstrating poorer performance.

Figure 4.54 shows the eCDF of throughput for the same two bandwidth throttles (6 Mbps and 8 Mbps) on the 3Pi testbed. Many crossing points signify a throughput limitation for the higher noise condition (6 Mbps). Interestingly, there are periods where the higher noise condition (6 Mbps) outperforms the lower noise condition (8 Mbps) e.g. after the 4.5 Mbps crossing point.

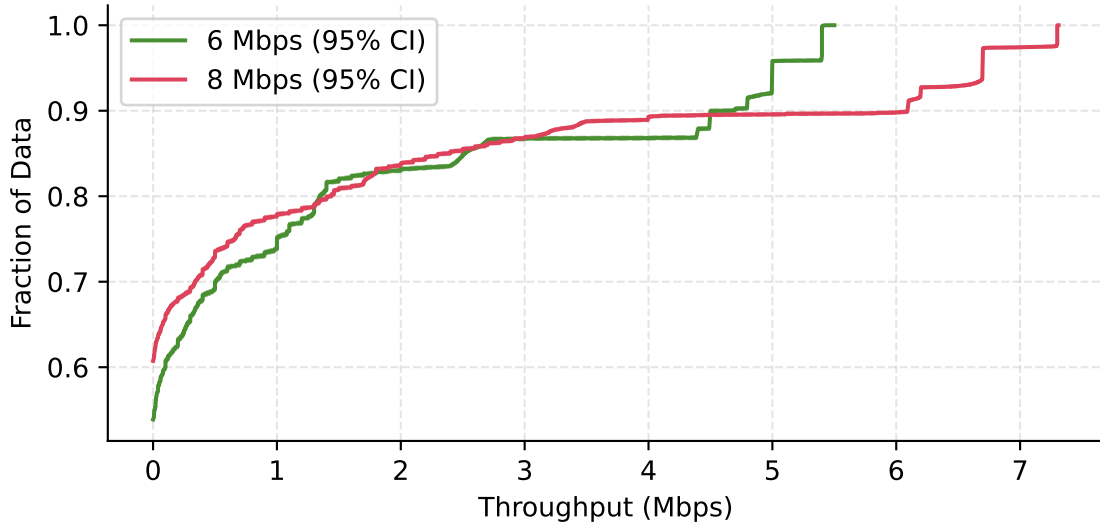


Figure 4.54: eCDFs with 95% confidence intervals of throughput (Mbps) for 6 Mbps (green) versus 8 Mbps (red) bandwidth throttle on 3Pi testbed.

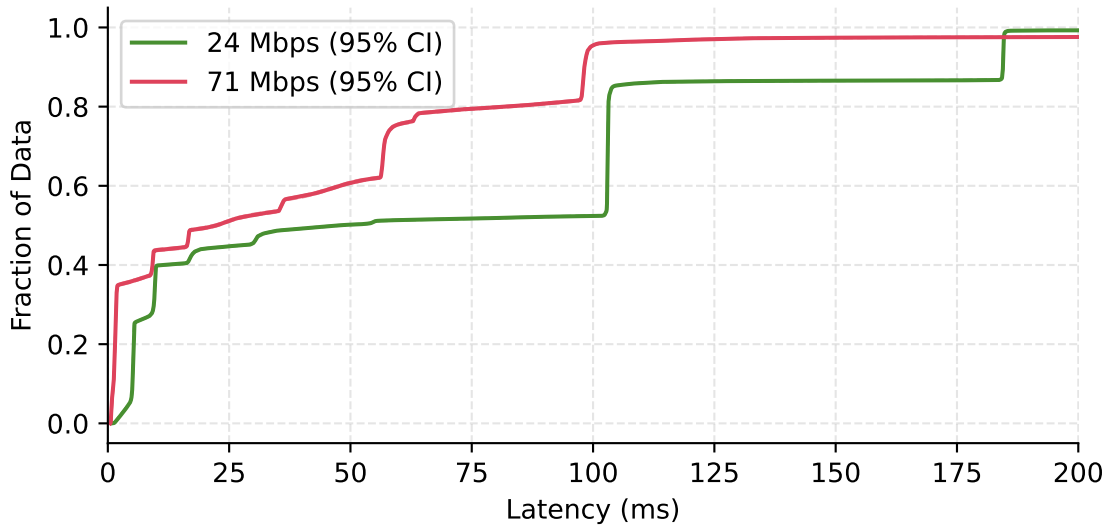


Figure 4.55: eCDFs with 95% confidence intervals of latency (ms) for 71 Mbps (red) versus 24 Mbps (green) bandwidth throttle on 5Pi testbed.

4.5.2 5 Raspberry Pi Analysis

Figure 4.55 shows the eCDF of latency for two different bandwidth throttles (71 Mbps and 24 Mbps as detailed in Subsection 3.4.5) on the 5Pi testbed. Reduced noise levels always result in lower latencies (except after 185 ms). The gap is especially pronounced for latencies between 50 ms and 110 ms. Overall, these results show that lower network noise leads to improved latency performance in the 5Pi testbed.

Figure 4.56 shows the eCDF of throughput for the same two bandwidth throttles (71 Mbps and 24 Mbps) on the 5Pi testbed. A substantial difference is observed following the initial crossover point. The lower noise condition (71 Mbps) exhibits a greater proportion of zero

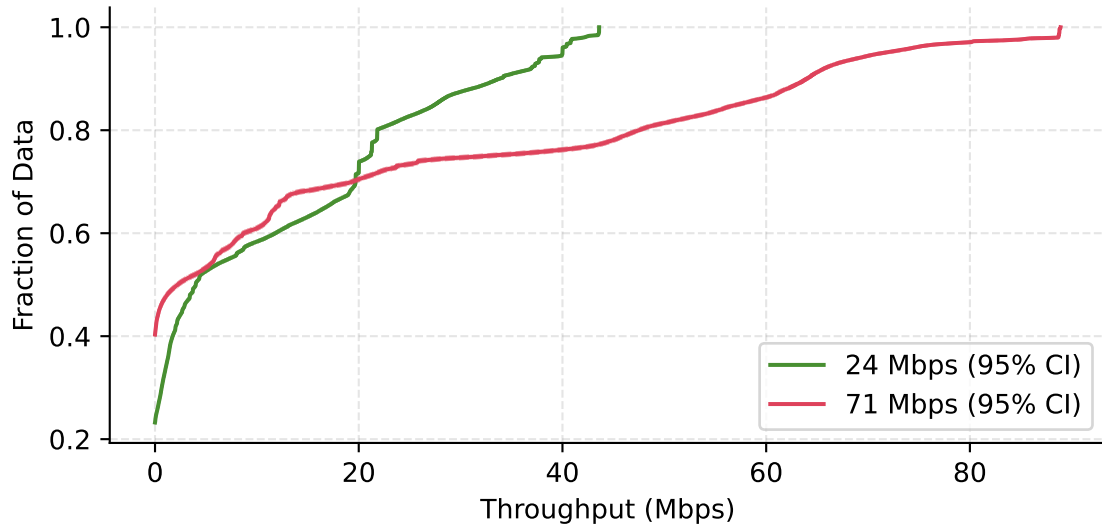


Figure 4.56: eCDFs with 95% confidence intervals of throughput (Mbps) for 71 Mbps (red) versus 24 Mbps (red) bandwidth throttle on 5Pi testbed.

throughput values, with approximately 40% of the data registering as 0 Mbps, whereas the higher noise condition (24 Mbps) shows fewer zero values, accounting for roughly 20% of the data. However, this observation pertains solely to the data preceding the initial crossover point. The introduction of noise results in a smaller proportion of data exhibiting 0 Mbps. Additionally, it leads to significantly lower throughput values subsequent to the initial crossover point. This suggests that while noise might initially reduce instances of zero throughput, it ultimately results in a substantial degradation of overall throughput performance beyond a certain threshold. The findings highlight a trade-off where a noisier environment might prevent complete communication stalls but at the cost of significantly reduced data rates for the majority of the communication.

4.5.3 Summary

This section analysed the impact of network noise on DDS performance, specifically focusing on latency and throughput in 3Pi and 5Pi testbeds. In the 3Pi testbed, higher bandwidth throttling (8 Mbps vs 6 Mbps) consistently led to increased latency, demonstrating poorer performance. For throughput, multiple crossing points indicated that higher noise conditions could sometimes outperform lower noise conditions after a certain threshold. In the 5Pi testbed, lower noise (71 Mbps vs 24 Mbps) resulted in reduced latency (under 185 ms). Throughput analysis revealed that while lower noise initially had more zero throughput values, introducing noise led to a smaller proportion of 0 Mbps data and significantly lower throughput after an initial crossover point.

4.6 Analysis of Faulty Throughput Measurements

During the performance evaluation of DDS, a number of experiments reported a throughput of 0 Mbps, even though data was being successfully sent by the publisher. These occurrences, while not frequent, are significant because they indicate a complete failure of data delivery from the perspective of the throughput measurement tool. This section presents an analysis of the experiments that resulted in a throughput of 0 Mbps. Understanding the causes of these faulty throughput measurements is crucial for identifying potential issues in the experimental setup, the DDS configuration, or the underlying network.

4.6.1 Analysis

This analysis was initiated due to the observation of a substantial proportion of 0 Mbps throughput values across all experiments within specific datasets. The analysis commenced by identifying which experiments specifically yielded 0 Mbps within each dataset. Subsequently, intersecting experiment names across datasets were examined. The investigation sought to determine if particular experiments consistently produced 0 Mbps, irrespective of their originating dataset. The findings indicated that this was not precisely the case. A further step involved analysing the configuration parameter values within the 0 Mbps experiments to identify any common themes, particularly within the 33 intersecting experiments observed in both the 3Pi and 5Pi datasets. The most frequently occurring configuration parameters were: 1 KB, 1 publisher, best effort, and UC.

Subsequently, the raw throughput values for each experiment were inspected. For each dataset, 9 random 0 Mbps experiments were sampled, and their raw values were plotted. It was concluded that the reliable experiments yielded meaningful 0 Mbps measurements, whilst best effort experiments generated a significant proportion of 0 Mbps values.

The investigation was then directed towards understanding why best effort configurations produced numerous 0 Mbps values. It was hypothesised that the 0 Mbps values were not genuinely 0 but rather rounded down, a suspicion confirmed upon inspection of the RTI's PerfTest source code. A comment within the source code further elucidated that employing best effort offers no control over flow or sending rate, which can lead to network flooding and CPU exhaustion. This comment explains the behaviour observed in our data. The data indicates network flooding, as the subscriber records minimal throughput, which is subsequently rounded to 0. However, during periods when the network processes messages accordingly, there are brief windows where the subscriber receives a small amount of data, appearing as visual

spikes in the raw data. This behaviour is corroborated by both the comment and the source code itself.

The analysis of the 0 Mbps throughput measurements has provided valuable insights into the behaviour of RTI's PerfTest under specific configurations. The findings suggest that the 0 Mbps measurements are primarily a result of the best effort configuration, which does not guarantee data delivery and can lead to network flooding. This flooding can overwhelm the network, resulting in the subscriber receiving no data, which is then recorded as 0 Mbps throughput. The analysis highlights the importance of understanding the underlying mechanisms of RTI's PerfTest configurations and their impact on performance metrics.

4.7 Limitations of the Experimental Setup

It is important to acknowledge the limitations of the experimental setup used in this study, as they may influence the generalisability of the findings.

The physical testbeds, while providing a controlled environment, are based on RPi clusters. While these devices are representative of many resource-constrained embedded systems, they do not reflect the performance characteristics of high-end, industrial-grade hardware. The results obtained on these testbeds may not be directly transferable to systems with more powerful CPUs, more memory, and faster network interfaces. However, there are still many use cases where DDS is deployed on resource-constrained devices, making these findings relevant for such scenarios.

The virtualised testbed, while representing a realistic enterprise environment, was a shared infrastructure. This means that the network was subject to noise and contention from other VMs and services, which could have influenced the performance measurements. While this provides a degree of realism, it also makes it harder to isolate the impact of specific DDS parameters.

The choice of DDS implementation, RTI Connex DDS, also has an impact on the results. Different DDS implementations may have different performance characteristics, and the findings of this study may not be directly applicable to other implementations.

Finally, the experimental campaigns, while extensive, could not cover the entire DDS configuration space. The choice of parameters and their ranges was necessarily a subset of all possible configurations. Therefore, the findings of this study should be considered as a snapshot of DDS performance under a specific set of conditions, rather than a comprehensive characterisation of all possible behaviours.

Despite these limitations, this study provides a valuable and detailed analysis of DDS performance under a wide range of conditions. The findings offer important insights into the complex interplay of factors that govern DDS performance and provide a solid foundation for future research in this area.

4.8 Conclusion

This chapter has presented a comprehensive experimental performance evaluation of the DDS. The study has systematically investigated the impact of various configuration parameters on the performance of DDS, focusing on latency and throughput. This final section summarises the main conclusions of the chapter.

The results of this study have demonstrated that the performance of DDS is highly dependent on the choice of QoS parameters and the characteristics of the underlying network. The reliability QoS setting was found to be a critical factor in determining the trade-off between reliability and performance. Multicast was shown to be an effective mechanism for improving scalability, but its performance is sensitive to network conditions. The data length of messages was also found to have a significant impact on both latency and throughput. The analysis of faulty throughput measurements highlighted the importance of careful experimental design and data analysis. In conclusion, this chapter provides a detailed and systematic performance evaluation of DDS that can be used to inform the design and deployment of real-time distributed systems.

Chapter 5

Model-Based Performance Prediction of Data Distribution Service

5.1 Introduction

DDS systems require optimal parameter settings to achieve peak performance and avoid delays or data loss. Randomly selecting parameter values for a specific scenario often leads to suboptimal results due to the complexity of the parameter space. Manually testing a few configurations and choosing the best is inefficient and unreliable. A systematic approach is needed to sample the parameter space and identify parameter values that maximise performance.

Exhaustive testing of all possible parameter combinations is computationally infeasible due to limited time and computational resources. The challenge is to find optimal or near-optimal parameter values that deliver high performance without testing every configuration. Sampling a subset of the parameter space and using the data to predict performance for untested configurations offers a practical solution.

This study proposes sampling parameter configurations and using ML models to predict performance. The sampled data trains a model to capture the relationship between input parameters and output performance distributions characterised by percentiles. This approach reduces the computational burden while aiming to identify high-performing parameter values.

ML provides several models suitable for this task. Random Forest (RF) (Breiman 2001) are robust and effective for modeling non-linear relationships. Extreme Gradient Boosting (XGB) (Chen & Guestrin 2016) is a powerful gradient boosting algorithm known for its high performance and efficiency. Feed-Forward Neural Network (FFNN) (Rumelhart et al. 1986) offer flexibility in capturing complex parameter-performance interactions. Linear Regression

(LR) (Gauss n.d.) (Kenney & Keeping n.d.) serves as a baseline to compare the predictive accuracy of these advanced models.

The proposed method involves sampling the parameter space and collecting performance data through DDS performance tests. This data is used to train and validate the ML models which are evaluated based on their prediction accuracy and computational efficiency. The best-performing model is then used to predict optimal parameter values.

The input consists of six configurable parameters; data length, publisher count, subscriber count, reliability, MC, and durability. Performance is evaluated using latency and throughput, which are the two most important metrics for DDS performance. Each parameter configuration generates a distinct distribution for these metrics and by analysing various statistical measures, these distributions can be characterised comprehensively.

For each distribution we can create a list of statistics that identify the distribution:

- mean
- standard deviation
- 21 percentiles: 0, 1, 2, 3, 5, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 95, 97, 98, 99, 100

We focus on the head of the distribution to capture worst-case throughput performance using the 0th (minimum), 1st, 2nd, 3rd and 5th percentiles and we focus on the tail of the distribution to capture worst-case latency performance using the 95th, 97th, 98th, 99th, and 100th (maximum) percentiles.

The work in this chapter can be split into two parts. In the first part of the work, we introduced the concept of using ML to predict DDS performance, focusing on LR and RFs. This work is the first of its kind and was published in (Peeroo et al. 2024). The second part is adding on to this work by introducing a new testbed (5Pi), some realism as network noise via varied cap levels on the bandwidth, 2 new models (XGB and FFNNs), a new subset of percentiles (23 percentiles instead of 21), a new approach to what is output from the model, and a deeper analysis into various modelling techniques.

The chapter is structured as follows:

- Section 5.1 introduces the problem and the motivation as well as provides a background on the knowledge for using ML to predict DDS performance.
- Section 5.2 reviews related work in the field of performance prediction using ML, particularly in the context of DDS and configurable systems (systems that can be configured with different parameters to achieve different performance characteristics).
- Section 5.3 describes the datasets used in the experiments, including the two testbeds

and the noise levels introduced and described in Chapter 3.

- Section 5.4 outlines the methodology for sampling the parameter space, collecting performance data, and training the ML models.
- Section 5.5 presents the hypotheses that will be tested in the experiments.
- Section 5.6 details the results of the experiments, including the performance of different ML models, the impact of network noise, and the effectiveness of linear versus log space modelling.

5.1.1 Contributions

Our contributions can be summarised as follows:

- **Initial Application of ML for DDS Performance Prediction:** The use of ML to predict DDS performance for the first time ever by modelling with LR and RFs as published in (Peeroo et al. 2024).
- **Extended Application with Additional Models:** We extend the initial work by incorporating XGB and FFNNs into the performance prediction framework alongside automated hyperparameter tuning to better capture a wide variety of modelling techniques.
- **6 New Datasets with Noise Levels:** We introduce 6 new datasets across 2 testbeds (3Pi and 5Pi) with 3 noise levels (no, low, and high) to simulate additional real-world network conditions characterising the performance of DDS in terms of latency and throughput using 23 percentiles.
- **Comprehensive Evaluation of Modelling Approaches:** We conduct an extensive evaluation of various modelling approaches including:
 - Interpolation versus extrapolation capabilities of the models to evaluate their generalisation to unseen parameter configurations.
 - Linear versus log space modelling to determine the most effective approach for capturing performance characteristics.
 - Single versus combined output modelling to assess the effectiveness of predicting both latency and throughput together versus separately.

5.1.2 Background

Machine Learning

ML, a branch of Artificial Intelligence (AI), concentrates on crafting algorithms and models that empower computers to learn from data and make informed decisions without specific programming directives. This concept revolves around the notion that computers can autonomously enhance their task performance by gleaning experience from data. ML encompasses a variety of techniques, including supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training models on labelled datasets, where the input data is paired with corresponding output labels. We use supervised learning in this work, as we have labelled data with the performance metrics for each parameter configuration. Compared to statistical modelling methods, ML models can capture complex, non-linear relationships directly from the data, making them particularly suitable for tasks like performance prediction in DDS systems. It specialises in identifying patterns and relationships within data, enabling the development of predictive models that can generalise to unseen data. This is something that statistical modelling methods struggle with, as they often rely on assumptions and a lot of contextual knowledge.

Linear Regression

LR is a fundamental statistical technique for modelling the relationship between a dependent variable (the target) and one or more independent variables (features). In our work, we chose LR primarily as a baseline model. Its simplicity, speed, and high interpretability provide a crucial benchmark for evaluating more complex models. By establishing a baseline, we can quantify the performance gains achieved by advanced techniques like RFs and FFNNs. The model assumes a linear relationship between DDS parameters and performance, and its coefficients offer a clear, albeit simplified, view of how each parameter influences the outcome.

Random Forests

RF is a powerful ensemble learning method chosen for their ability to model complex, non-linear relationships, which are highly anticipated in DDS performance data. The model builds multiple decision trees on different subsets of the data and features, averaging their predictions to improve accuracy and control overfitting. We selected RFs for several key reasons: their proven effectiveness in similar performance prediction tasks, their robustness against noisy data, and their inherent ability to calculate feature importance. This last attribute is particularly

valuable in our context, as it helps identify which DDS parameters most significantly impact performance, offering deeper insights beyond simple prediction.

XGBoost

XGB is an advanced implementation of gradient boosting, selected for its state-of-the-art performance and efficiency. Unlike RFs which build trees independently, XGB builds them sequentially, with each new tree correcting the errors of the previous one. We chose XGB to see if this more sophisticated ensemble technique could yield higher predictive accuracy than RFs. Its key advantages include high performance, built-in regularisation to prevent overfitting, and its ability to handle missing data, making it a robust choice for our complex and potentially noisy DDS performance dataset.

Feed Forward Neural Networks

FFNNs are powerful computational models capable of learning extremely complex and subtle patterns in data. We included them in our study because of their potential to capture intricate, non-linear interactions between DDS configuration parameters that other models might miss. Their layered structure allows them to learn features at different levels of abstraction, making them highly flexible and powerful for regression tasks. While often considered a “black box” due to their lower interpretability, their potential for achieving the highest predictive accuracy makes them an essential model to evaluate for the task of DDS performance prediction. A limitation of FFNNs is that they require more data to train effectively compared to other models, which can be a challenge in scenarios with limited data availability. Nonetheless, their ability to model complex relationships makes them a valuable addition to our analysis.

Standardisation

Standardisation is a preprocessing technique that transforms data to have a mean of zero and a standard deviation of one. It scales features to a common range, ensuring comparability across variables with different units or scales. By centering and normalising data, standardisation improves the performance and convergence of ML algorithms, such as gradient-based methods, that are sensitive to feature magnitude. It is widely used in tasks like regression, classification, and clustering to enhance model accuracy and stability. We apply standardisation to our datasets to ensure that all features contribute equally to the model training process, preventing any single feature from disproportionately influencing the results due to its scale.

Transformation

Transformations are preprocessing techniques applied to data to modify its distribution, scale, or structure, enhancing suitability for ML algorithms. Common transformations include normalisation, which scales data to a fixed range (e.g., $[0, 1]$), and logarithmic or power transformations, which stabilise variance or address skewness. These methods improve model performance by ensuring data meets algorithmic assumptions, such as linearity or normality, and are widely used in tasks like regression, classification, and clustering to optimise accuracy and convergence. We apply transformations to our datasets to ensure that the data is in a suitable format for the ML algorithms, particularly when dealing with skewed distributions or varying scales across features. Latency and throughput data often exhibit skewed distributions, making transformations necessary to improve model performance and interpretability.

Model Evaluation

Numerous metrics are available for assessing the performance of a model. In our study, we have chosen to concentrate specifically on two of these metrics: Root Mean Square Error (RMSE) and Coefficient of Determination (R^2). We selected these metrics because they provide complementary insights into model performance. We use RMSE to quantify the average magnitude of prediction errors, offering a clear measure of how close the predicted values are to the actual values which is particularly useful for latency measurements with a granularity of microseconds. We use R^2 to assess the proportion of variance in the dependent variable that is explained by the independent variables, providing a measure of how well the model fits the data. The R^2 also lets us compare the performance of different models, as it is a normalised standard metric in regression analysis.

Root Mean Squared Error

The RMSE measures the average size of the errors or residuals between predicted and actual observed values. The formula for RMSE is given by:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_{\text{predicted},i} - y_{\text{actual},i})^2}$$

This calculation involves taking the square root of the average of the squared differences between the predicted and actual values, as demonstrated in the formula above. RMSE is expressed in the same units as the dependent variable, enhancing its interpretability in relation to the problem at hand. A lower RMSE value signifies a more accurate model, as it indicates that

the predictions are closer to the actual observations. However, RMSE is particularly sensitive to outliers, as the squaring of errors disproportionately emphasises larger discrepancies, which can markedly affect the RMSE value.

Coefficient of Determination

The R^2 , is a statistical measure used in regression analysis to assess the fit quality of a model. It usually ranges between 0 and 1, indicating the fraction of the variance in the dependent variable that is accounted for by the independent variables. The R^2 formula is as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_{\text{actual},i} - y_{\text{predicted},i})^2}{\sum_{i=1}^n (y_{\text{actual},i} - \bar{y})^2}$$

This metric can be interpreted as the ratio of the explained variance (the sum of squares of the fitted values) to the total variance (the sum of squares of the actual values). A higher R^2 value signifies that a greater portion of the variance in the dependent variable is explained by the model, implying a better fit. However, it is important to note that R^2 does not convey the direction or significance of the relationship. Even with a high R^2 , scrutinising the individual coefficients and the context of the data remains crucial, which is why RMSE is also considered in our analysis.

5.2 Related Work

This study pioneers the application of ML models for predicting DDS performance. A thorough literature review found no direct precedents in this specific area. However, expanding the scope of our search revealed relevant studies under broader terms in related fields.

The first pertinent field is *Performance Prediction of Distributed Systems*. Here, studies like (Bouloukakis et al. 2018) have used Stochastic Petri Net (SPN)s for assessing the performance of publish-subscribe systems, while (Sachs et al. 2013) applied them in the context of message-oriented middleware.

The second relevant area is *Performance Prediction of Configurable Systems using ML*. Numerous studies have focused on predicting the performance of systems with adjustable parameters, where different configurations impact overall performance. These investigations predominantly leverage ML techniques, but typically they forecast a *single* statistical figure representing a chosen performance metric, such as the mean value.

In works like (Bao et al. 2018), an RF-based comparative model is employed to identify the optimal system configuration for maximising throughput. (Guo et al. 2018) and (Siegmond et al.

2015) integrate ML methods and sampling strategies to predict performance in configurable systems. These strategies select training data based on configuration positions within a multi-dimensional space defined by the configuration options’ values. (Ha et al. 2019a) employs a deep FFNN to predict performance across different configurations, representing the performance with a singular value. An innovative approach is seen in (Shu et al. 2020), where adversarial learning via Generative Adversarial Network (GAN) is applied to predict and optimise performance with fewer samples than traditional techniques. Moreover, (Zhang et al. 2015) and (Ha et al. 2019b) explore performance prediction in systems with solely Boolean configuration options.

5.2.1 Our Contributions in Context

Our work significantly advances the field by addressing several key limitations and expanding upon existing methodologies: This study is the first to systematically apply ML models to predict DDS performance. Unlike traditional methods such as SPNs used in (Bouloukakis et al. 2018) and (Sachs et al. 2013), our approach leverages data-driven ML to capture complex, non-linear relationships inherent in DDS systems.

While prior work in configurable systems typically forecasts a *single* statistical figure (e.g., mean value as in (Ha et al. 2019a)), our research innovates by predicting comprehensive performance distributions, characterised by multiple percentiles. This provides a richer understanding of performance, especially for critical worst-case scenarios.

Building on the survey by (Grebhahn et al. 2019) which advocates for RF and LR, we extend our analysis to include XGB and FFNN models. Our work provides a deep, systematic comparative analysis of these models under various realistic conditions, including different datasets, noise levels, transformations, and single versus combined output.

We introduce and analyse the impact of network noise through bandwidth caps, a novel approach that addresses real-world complexities in DDS deployments. Our findings, such as the surprising regularising effect of moderate noise, offer new insights into model robustness in noisy environments.

We conduct a rigorous analysis of interpolation and extrapolation, revealing that tree-based models can perform effectively in extrapolation under specific conditions, challenging conventional assumptions about their generalisation capabilities.

We thoroughly investigate the effects of linear versus logarithmic transformations on model performance for different percentile scopes and metrics, providing practical guidance on data preparation for DDS performance prediction.

Our comparison of single versus combined metric prediction explores a novel aspect of performance modeling. Unexpectedly, combined output models can sometimes outperform single output models, suggesting exploitable underlying correlations between metrics.

Our work draws substantial inspiration from (Rausch & Sanders 2020), which mirrors our process conceptually. The ML methodologies we adopted are based on insights from (Grebhahn et al. 2019), a comprehensive survey reviewing ML techniques and sampling strategies for performance prediction in configurable systems. This survey advocates using three primary ML methods: RFs, multiple LR, and classification and regression trees.

5.3 Datasets

To comprehensively evaluate the performance prediction models, we collected data across six distinct datasets. These datasets are derived from combinations of two different testbeds and three varying bandwidth cap levels, designed to simulate diverse operational environments and network conditions.

5.3.1 Bandwidth Caps

To introduce varying levels of network realism, we applied different bandwidth caps during data collection:

- **No Cap:** Represents an ideal network environment with no artificial bandwidth limitations, serving as a baseline for optimal performance.
- **High Cap:** Introduces a high cap on the bandwidth, simulating a network with some congestion or limited capacity. This level helps assess model robustness under typical, non-ideal conditions.
- **Low Cap:** Imposes a significant bandwidth restriction, mimicking highly constrained or noisy network environments. Data from this setting is crucial for evaluating model performance in challenging, real-world scenarios where network resources are scarce.

Each dataset captures the performance of DDS configurations defined by six configurable parameters: data length, publisher count, subscriber count, reliability, MC, and durability. By varying these parameters across the different testbeds and bandwidth caps, we generated a rich and diverse collection of performance data, enabling a thorough evaluation of our ML models' generalisability and predictive accuracy. Summaries of the datasets are presented in Table 5.1 which shows the number of samples (experiments) collected for each testbed and bandwidth

cap, along with the percentage of samples truncated for latency and throughput metrics based off the data processing mentioned in Section 3.5.

Testbed	Bandwidth Cap	Samples	Latency Truncation (%)	Throughput Truncation (%)
3PI	No	1537	15	20
	6 Mbps	339	10	40
	8 Mbps	352	10	40
	3PI Total	2228		
5PI	No	549	10	20
	24 Mbps	629	10	30
	71 Mbps	610	20	30
	5PI Total	1788		
	Total	4016		

Table 5.1: Summary of the number of samples per testbed and bandwidth cap.

5.4 Method

We are likely to measure more samples using performance tests that use lower load configuration parameter values e.g. smaller data length, single publisher and subscriber, and no reliability compared to performance tests that use higher load configuration parameter values e.g. larger data length, multiple publishers and subscribers, and reliability. This is due to the fact that higher load configurations are more likely to cause delays which results in fewer samples being measured. To get a better understanding of the system performance we would like as many samples as possible. However, if we are interested in configurations for high load scenarios, we need to sample the parameter space in such a way that we get enough samples for high load configurations. What if we could run performance tests on lower load configurations and then use the data to predict the performance of higher load configurations? To test this out, we split the dataset into two parts: inside, and outside. Inside consists of the configurations that use lower load parameter values (e.g. 1 publisher and 1 subscriber) and outside consists of the configurations that use higher load parameter values (e.g. 100 publishers and 200 subscribers). The models will train, test, and validate on the inside dataset and then predict the performance of the outside dataset (as well as predict on the inside because of the validation set).

5.4.1 Dataset Processing

When predicting the performance of a system, it is usual to predict a single value; the average. In some cases, researchers also predict the standard deviation or variance. However, our aim is to best capture the performance through its entire distribution. One method is to predict a single value, as mentioned, while another method is to predict a distribution function which is non-trivial. We choose a method that balances the two by choosing to predict a set of percentiles for each metric that are used to recreate the distribution. Models theoretically perform worse in the case where there are more values to predict. Therefore, we have chosen to recreate the distributions according to a subset of percentiles, striving to keep a balance between enough data to recreate the distribution while also producing accurate enough predictions. In both parts of the work, we process the data into a single dataset where each row represents a single experiment and the metrics for each experiment is represented by a list of percentiles. These statistics include: mean, standard deviation, and the following quantiles:

0, 1, 2, 5, 10

20, 25, 30, 40, 50, 60, 70, 75, 80

90, 95, 99, 100

We extended this in the second part of the work by adding the 3%, 97%, and 98% percentiles to capture the tail and head of the distribution better. Capturing the tail and head of the distribution ensures that we have the worst-case performance for both latency (highest values) and throughput (lowest values). Therefore, for each metric column produced from the previous data processing steps, we gathered the mentioned statistics. All data were combined with the parameter settings used for the experiment including the duration in seconds, data length in B, publisher count, subscriber count, reliability, MC, and durability. All Boolean columns e.g. reliability were parsed into integers with 0 representing False and 1 representing True and categorical columns e.g durability had one-hot encoding applied to them meaning that the durability column would be split into 4 columns: `durability_0`, `durability_1`, `durability_2`, `durability_3`. Each durability column would take a Boolean value indicating which durability level was used. So each row in the dataset is made up of the input parameters and the output metric percentiles. The input parameters being the duration in seconds, data length in B, publisher count, subscriber count, reliability, MC, `durability_0`, `durability_1`, `durability_2`, `durability_3` and the output metrics being the 21 percentiles for latency and 21 for throughput. In total each row has 11 input parameters and 42 output metrics.

5.4.2 Bandwidth Cap Level Comparison

For the second part of the work, we extended the previous work by introducing BW cap levels and the new 5Pi testbed. This introduced 5 new datasets:

- 3Pi
 - No Cap (from first part of the work)
 - 6 Mbps
 - 8 Mbps
- 5Pi
 - No Cap
 - 24 Mbps
 - 71 Mbps

Some experiments executed successfully for some cap levels while not failing for other levels. There was an inconsistency between the number of successfully executed experiments for different cap levels. We want to see how the models handle “realism” through the added “noise” and we need to make sure the datasets share the same experiments in an ideal scenario. However, some experiments were not successful (probably due to the introduced cap levels). Lower cap levels meant less BW and experiments that used many participants with larger data lengths would often saturate the network leading to major delays and resulting in experiments not finishing due to these delays. Ensuring that all experiments were successful across all cap levels would take greater effort and time than provided. Therefore, we opt to use the data that we were able to successfully collect across all cap levels. To make sure that the models are trained on the same experiments across all cap levels, we had to trim down the datasets to match the smallest dataset. This results in all datasets having the same number of experiments (339 for 3Pi and 549 for 5Pi) across all cap levels which means that we can directly compare the same experiments across all cap levels ensuring fairness.

5.4.3 Dataset Allocation Strategy

There are two techniques for predicting values: interpolation and extrapolation. Interpolation involves the estimation of values within the range of known data points whilst extrapolation involves the estimation of values outside of the range. Ideally, the perfect scenario would be a model that can accurately extrapolate. This would mean that one could collect data of configurations that are not too resource-intensive to predict configurations that could be too resource-intensive. Extrapolation can, in principle, work in the other direction: to extrapolate

the less resource intensive configurations. However, for DDS practioners, this is less interesting because they really want to capture the “load function” with use of extrapolation. Therefore, we had to explicitly define the input parameter values that we would consider for interpolation and extrapolation.

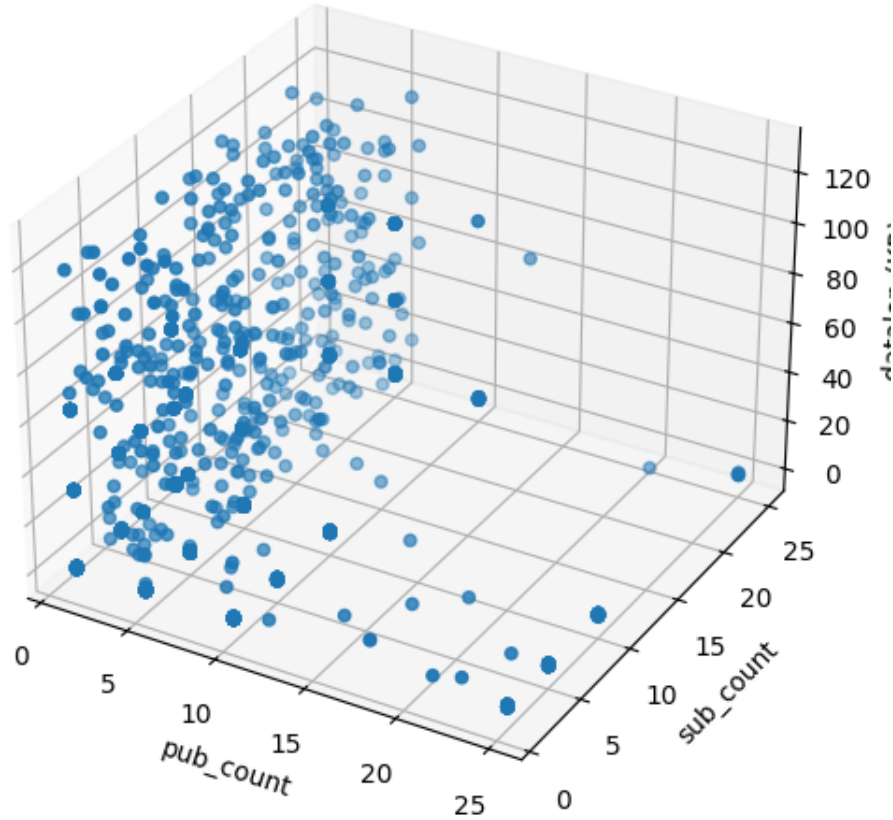


Figure 5.1: 3D scatter plot of the values for publisher/subscriber count and data length (B).

We created a 3D scatter plot seen in Figure 5.1 to visualise the configuration space using the only three numerical parameters of the six: publisher count, subscriber count, and data length. We also generated three 2D scatter plots for each pair of parameters as shown in Figure 5.2 to better understand the sampling space.

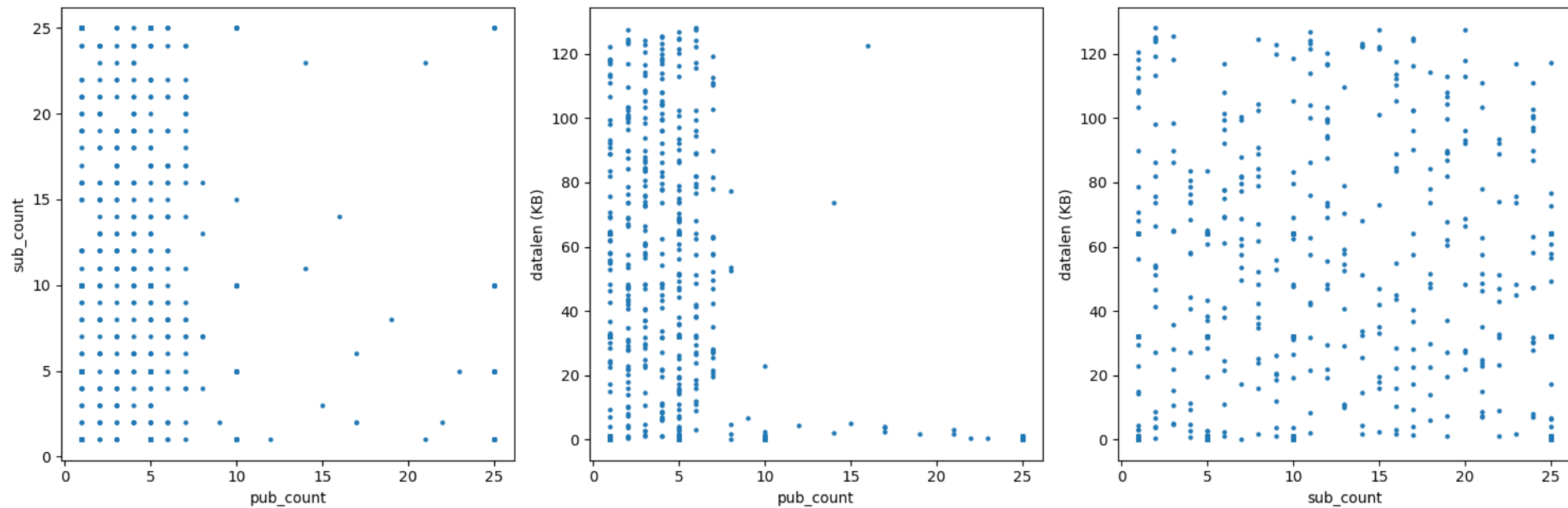


Figure 5.2: All variations of the 3D scatter plot as seen in 2D.

Ideally, the data should be split into three sections: training, testing for interpolation, and testing for extrapolation where the number of samples used for testing for interpolation and extrapolation should be the same. To define the range that also fits with the 80-20 split, we started off by defining values for the three parameters as shown below:

- Publisher count 5, 10, 15, 20, 25
- Subscriber count 5, 10, 15, 20, 25
- Data length (B) 40,000, 60,000, 80,000, 100,000, 120,000

The Cartesian product would then be applied to all three lists, producing all possible combinations of values for the three parameters. For each combination, the number of experiments with values lower than or equal to the chosen values were counted as well as the opposite case: tests with values greater. The inside and outside percentages were also calculated to find the best match of 80% inside and 20% outside. There were over a hundred different combinations with various inside and outside percentages. The closest percentage to the 80-20 split is the combination of 10 publishers, 25 subscribers, and 80 KB for the data length and this is where we draw the line to define the range of parameter values for interpolation and extrapolation. The final split with the values carefully chosen to balance the allocation of data whilst being as close to the middle value as possible, is visually shown in Figure 5.3 showing the values of the three parameters for the boundary, the number of experiments inside (green), the number of experiments outside (blue) and the percentage of experiments inside and outside.

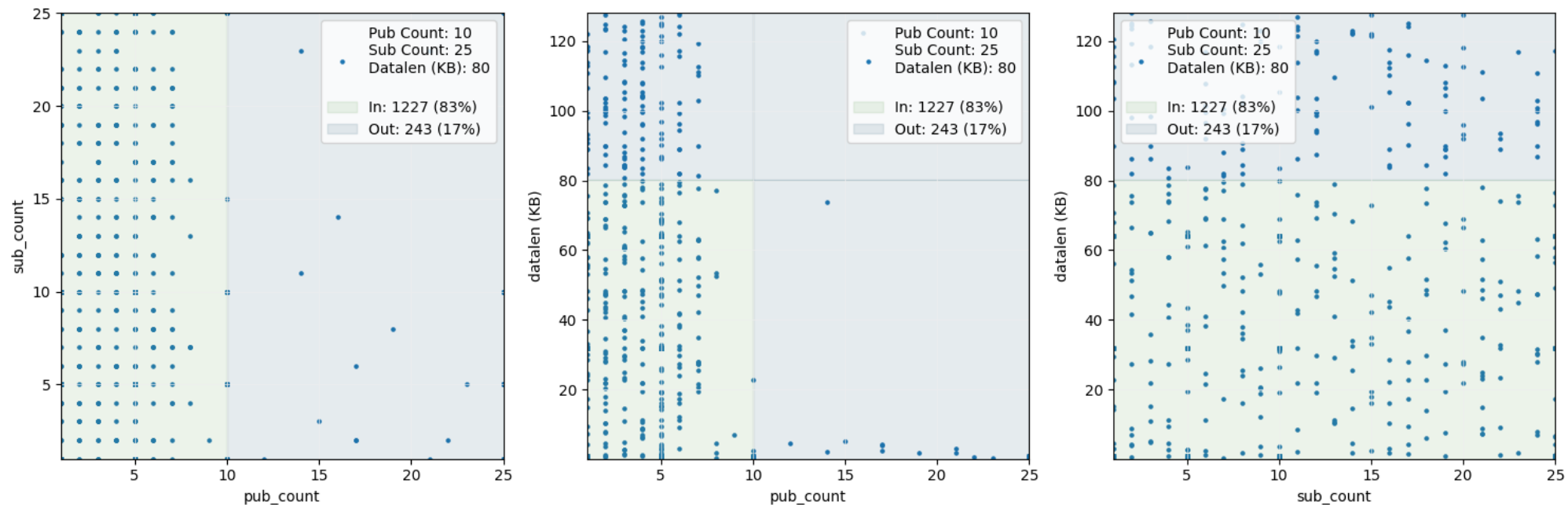


Figure 5.3: All variations of the 3D scatter plot as seen in 2D (green area indicates values used for interpolation and blue area indicates values used for extrapolation).

This concludes how we split the data into inside and outside sections for the published part of the work. This was done on the no noise variation dataset on the 3Pi testbed and had to be replicated for the other datasets. We extended this process by automating the allocation of parameter values marking the boundary for interpolation and extrapolation. On top of this, we also introduced a validation set (from the inside section) to be able to validate the model after training which resulted in the following split:

- Training (Inside): 56%
- Testing (Inside): 12%
- Validation (Inside): 12%
- Extrapolation (Outside:) 20%

To automate the process of splitting the data into inside and outside sections, we created a function that takes in a dataset and returns the inside and outside sections.

1. Split the dataset into two parts: inside and outside:
 - `split_in_out(df)` outputs:
 - `df_in`: Inside dataset (for interpolation)
 - `df_out`: Outside dataset (for extrapolation)
2. Split the dataset input and output columns:
 - `split_X_y(df_in)` outputs:
 - `df_in_X`: Input features for the inside dataset
 - `df_in_y`: Output features for the inside dataset
 - `split_X_y(df_out)` outputs:
 - `df_out_X`: Input features for the outside dataset
 - `df_out_y`: Output features for the outside dataset
3. Split the inside dataset into training, testing, and validation sets:
 - `split_train_test_val(df_in_X)` outputs:
 - `df_X_train`: Training set input features
 - `df_X_test`: Testing set input features
 - `df_X_val`: Validation set input features
 - `split_train_test_val(df_in_y)` outputs:
 - `df_y_train`: Training set output features
 - `df_y_test`: Testing set output features
 - `df_y_val`: Validation set output features

So overall, we take a dataframe `df` and split it into 8 sections:

- **df_X_train**: Training set input features
- **df_X_test**: Testing set input features
- **df_X_val**: Validation set input features
- **df_out_X**: Outside set input features (for extrapolation)
- **df_y_train**: Training set output features
- **df_y_test**: Testing set output features
- **df_y_val**: Validation set output features
- **df_out_y**: Outside set output features (for extrapolation)

To automate this, we replicate the process explained previously by defining the values for the three numerical parameters and using the Cartesian product to find the closest split to 80% inside and 20% outside. We then split the data inside (interpolation) and outside (extrapolation) into input features (X) and output features (y). To split the data inside (interpolation) into train, test, and validation sections, we shuffle the data, define the train, test, and validation size and cut up the data into the 3 sections accordingly. We define a train size of 70%, test size of 15% and validation size of 15% within the data inside. Looking at the entire dataset we have the following split:

- Training (Inside): 56% (calculated as 70% of 80%)
- Testing (Inside): 12% (calculated as 15% of 80%)
- Validation (Inside): 12% (calculated as 15% of 80%)
- Extrapolation (Outside): 20%

Tables 5.2 and 5.3 summarise the final chosen configuration parameter values used to split the datasets into inside and outside samples for 3Pi and 5Pi respectively along with the number of samples allocated to each section.

	No	Low (6 Mbps)	High (8 Mbps)
Publisher Count	28	10	20
Subscriber Count	75	15	15
Data Length (B)	64000	1000	1000
In Count	1202	266	285
Out Count	335	73	67
In (%)	78	78	80
Out (%)	22	22	20

Table 5.2: Summary of the configuration parameters values used to split the dataset into inside (used for interpolation) and outside (used for extrapolation) samples for 3Pi.

	No	Low (24 Mbps)	High (71 Mbps)
Publisher Count	24	20	15
Subscriber Count	28	15	20
Data Length (B)	61589	1000	1000
In Count	442	352	489
Out Count	107	97	121
In (%)	80	84	80
Out (%)	20	16	20

Table 5.3: Summary of the configuration parameters values used to split the dataset into inside (used for interpolation) and outside (used for extrapolation) samples for 5Pi.

5.4.4 Model Training

In the published part of the work (Peeroo et al. 2024), we trained models using z-score, min-max, robust, and no standardisation on the inputs and log, log10, log2, log1p, and square root transformations on the outputs for latency, throughput, lost samples, lost samples (%), received samples, received samples (%), and sample rate. LR was chosen as a baseline for comparison, whilst RF was the selected model to be used as the first step in using ML with DDS for performance prediction. For both models, we explored all possible combinations of input standardisations and output transformations. For interpolation, we used 1227 experiments for training (inside the boundary) and 245 experiments for testing (inside the boundary), while for extrapolation we used 243 experiments for testing (outside the boundary).

Models are trained on a single DDS performance metric, so there is one model per DDS performance metric e.g. sample rate. After allocating the training and testing datasets, the standardisation is applied to the inputs and the transformations to the outputs. The model is then trained and the error metrics are calculated on the predictions. It is critical to highlight that the error metrics are calculated for each statistic used to recreate the distribution of the metric, e.g. 5th percentile of latency. This means that all evaluation metrics mentioned later e.g. RMSE, are applied for each individual statistic, e.g. 60th percentile. We also average the error metrics across all statistics to get a single value for each metric. It must be noted that transformations are reversed before calculating the error metrics to ensure that the error metrics are calculated in the original space of the metric.

To extend this published work we added two new models: XGB and FFNN. We also realised that the combination of input standardisations, output transformations, models, and DDS performance metrics would result in a large number of models to be trained. In the interest of time, we focus on the latency and throughput metrics only. We also introduce a new method of predicting the performance by combining the two metrics into the same output consisting of

46 percentiles (23 for latency and 23 for throughput). Even with this reduction, we still have many variations:

- **4 models:** LR, RF, XGB, and FFNN
- **3 types of outputs:** latency, throughput, and both
- **2 testbeds:** 3Pi, 5Pi
- **3 BW cap levels:** No Cap, Low Cap (6 Mbps for 3Pi and 24 Mbps for 5Pi), High Cap (8 Mbps for 3Pi and 71 Mbps for 5Pi)
- **3 input transformations:** none, z-score, and minmax
- **2 output transformations:** linear and log

In total there are $4 \times 3 \times 2 \times 3 \times 3 \times 2 = 432$ different model variations. For each FFNN, RF, and XGB model we need to tune the hyperparameters and doing each one manually would take too long. Therefore, we use Optuna (Akiba et al. 2019) to automate this process. We run a study for each combination of model, metric (output type being latency, throughput or all), input transformation, and output transformation. The study optimises for minimal RMSE and are separated by output transformation so that RMSE calculated in linear space is never compared to RMSE calculated in log space. We run the studies for 5 minutes each resulting in a total run of $5 \times 324 = 1620$ minutes (27 hours). We multiply by 324 which is $\frac{3}{4} \times 432$ because the LR models have no hyperparameters to be tuned.

For each dataset, model, metric, and output transformation, the best input transformation and hyperparameters are chosen according to the lowest RMSE. We then take these values and train and run the model 10 times to produce a distribution allowing us a foundation for statistical analysis. For each repetition, we calculate the RMSE, R^2 , Mean Absolute Error (MAE), and Maximum Absolute Error (MaxAE) for each output variable. A complete rundown of the hyperparameter search space for each model can be seen in Table 5.4.

Model	Hyperparameter	Description	Search Range / Options
RF	n_estimators	Number of trees in the forest.	100 to 1000 (step 100)
	max_depth	Maximum depth of the tree.	3 to 10
	min_samples_split	Minimum samples required to split an internal node.	2 to 10
	min_samples_leaf	Minimum samples required to be at a leaf node.	1 to 10
	min_weight_fraction_leaf	Minimum weighted fraction of the sum total of weights.	0.0 to 0.5
	max_features	Number of features to consider for the best split.	None, "sqrt", "log2"
	bootstrap	Whether bootstrap samples are used when building trees.	True, False
XGB	n_estimators	Number of gradient boosted trees.	100 to 1000 (step 100)
	learning_rate	Step size shrinkage used in update to prevent overfitting.	0.01 to 0.3
	subsample	Subsample ratio of the training instances.	0.5 to 1.0
	colsample_bytree	Subsample ratio of columns when constructing each tree.	0.5 to 1.0
	max_depth	Maximum depth of a tree.	3 to 10
	alpha	L1 regularisation term on weights.	0.0 to 1.0
	reg_lambda	L2 regularisation term on weights.	0.0 to 1.0
FFNN	n_layers	Number of hidden layers.	1 to 3
	units_{i}	Number of units in layer i .	16 to 64
	learning_rate	Learning rate for the optimiser (log scale).	10^{-4} to 10^{-1}
	optimizer	Optimisation algorithm to use.	"adam", "sgd"
	batch_size	Batch size for training.	16, 32, 64
	dropout	Dropout rate.	0.0 to 0.5
	weight_decay	Weight decay (L2 regularisation, log scale).	10^{-6} to 10^{-2}

Table 5.4: Optuna hyperparameter search space for RF, XGB, and FFNN models.

Interpolation vs Extrapolation

This subsection aims to explain how each model handles interpolation and extrapolation differently based on their underlying mechanisms to give better context to the results presented later.

For interpolation, LR models use the global closed-form solution to find the best-fitting line that minimises the error between the predicted and actual values within the training data range. This means that for interpolation, LR can provide accurate predictions as long as the underlying relationship between the input features and the target variable is linear or can be approximated linearly. However, for extrapolation, LR models project the linear trend indefinitely beyond the range of the training data. Due to this assumed global functional form, it “boldly” predicts outside the training range, though the accuracy, much like with interpolation, rely on the linearity of the relationship between the input features and the target variable.

Interpolation with RF and XGB models is based on local averaging. The prediction is the mean of the target values in the leaf nodes of the ensemble of decision trees that correspond to the input features. This means that for interpolation, RF and XGB models can provide accurate predictions as long as the input features fall within the range of the training data. However, for extrapolation, these models struggle because they cannot predict values outside the range of the training data. When input features fall outside the training range, the model defaults to the nearest leaf nodes, leading to potentially inaccurate predictions that do not capture the true underlying trend. This results in the model “plateauing” at the boundary values of the training data. If the max training target was 100, the model could never predict anything higher than 100 regardless of the input features.

In regards to our work, if the tree-based models have not seen measurements that they were trained on, they will struggle to predict values outside of this range. However, we have chosen to split the data according to the input parameter values rather than the output metric values. This means that the models may have seen similar output metric values during training even if the input parameter values are outside the training range. This could help the models make better extrapolation predictions since they have seen similar output metric values during training. These models will struggle in cases where the output metric values are significantly different from those seen during training.

FFNN models act as a universal function approximator. By using non-linear activation functions and multiple layers, FFNN models can learn complex relationships between input features and target variables. Therefore, for interpolation, FFNN models can provide accurate

predictions as long as the input features fall within the range of the training data. However, extrapolation can be highly unpredictable since ReLU networks become linear functions at the extremes resulting in “overconfident” linear predictions that may not reflect the true underlying trend. On the other hand, saturating activation functions (e.g. sigmoid, tanh) will plateau, similar to tree-based models, as the neurons reach their limits.

5.4.5 Summary

In the interest of reproducibility, this summary aims to summarise the steps to reproduce the results presented in this work. This involves the process of going from raw data to processed datasets and finally to trained models and results.

1. **Dataset Processing** (Section 5.4.1):

- Process the raw data into a single dataset where each row represents a single experiment with input parameters and output metric percentiles.

2. **BW Cap Level Comparison** (Section 5.4.2):

- Trim down datasets to match the smallest dataset across all cap levels to ensure consistency when comparing across different noise levels.

3. **Dataset Allocation Strategy** (Section 5.4.3):

- Split the dataset into inside (interpolation) and outside (extrapolation) sections based on defined parameter values.
- Further split the inside dataset into training, testing, and validation sets.

4. **Model Training** (Section 5.4.4):

- Train models using various input standardisations and output transformations.
- Evaluate models using error metrics for each statistic used to recreate the distribution of the metric.
- Automate hyperparameter tuning using Optuna for RF, XGB, and FFNN models.

5.5 Hypotheses

In this section, we present the hypotheses that we will test in the ML experiments. Each hypothesis is associated with a specific aspect of the model’s behavior or performance. We start by outlining the primary hypothesis regarding the feasibility of using ML for predicting DDS performance. This is followed by hypotheses that explore the model’s performance in different scenarios, such as interpolation versus extrapolation, the impact of network noise, the

effects of linear versus log space modelling, and the comparison between single and combined output predictions. Only the first two hypotheses were tested in our published work and we have extended the discussion to include additional hypotheses that were not part of the original publication but are relevant to this work.

5.5.1 Feasibility of Machine Learning for Data Distribution Service Performance Prediction

Can ML models effectively predict DDS performance, and how do they compare against a simple baseline?

Our primary hypothesis is that ML is a viable and effective method for predicting the performance of complex, configurable systems like DDS. We further hypothesise that non-linear models will be essential for capturing the intricate relationships between configuration parameters and performance outcomes. Specifically, we expect that a sophisticated model like RF will significantly outperform a basic LR model. We anticipate that LR will struggle, resulting in low R^2 scores and high RMSE, thereby serving as a baseline to demonstrate the complexity of the problem. Conversely, we expect RF to yield high R^2 scores and low RMSE, validating the feasibility and utility of using ML for this task. This initial investigation seeks to answer the fundamental question of whether ML can model DDS performance with sufficient accuracy to be useful.

5.5.2 Interpolation Versus Extrapolation

How do the models handle extrapolation compared to interpolation?

We hypothesise that the models will perform significantly better at interpolation than extrapolation. This is because interpolation involves making predictions within the range of the training data, where the model has learned the underlying patterns. Extrapolation, on the other hand, requires the model to predict for configurations outside of its training experience, which is a much more challenging task. We expect to see higher RMSE and lower R^2 for the extrapolation tasks compared to the interpolation tasks. We are looking to quantify this performance degradation and identify which models are more robust to extrapolation.

5.5.3 Network Noise

How does the newly introduced “network noise” via bandwidth cap affect the modelling?

We hypothesise that the introduction of network noise, through bandwidth capping, will make the performance prediction task more difficult, leading to a decrease in model accuracy. The noise is expected to introduce more variability and unpredictability into the performance data. We expect that the models trained on datasets with no bandwidth cap will perform the best, while the models trained on datasets with a low bandwidth cap will perform the worst. We are looking to see how the different models handle this noise and whether some models are more resilient to it than others.

5.5.4 Linear Versus Log Space

How does modelling in linear space compare to modelling in log space?

We hypothesise that modelling in log space will improve the performance of the models, especially for latency which often has a skewed distribution. By transforming the target variables to a logarithmic scale, we can often normalise the distribution and reduce the influence of outliers. This should make it easier for the models to learn the underlying relationships. We expect to see lower RMSE values for the models trained in log space compared to those trained in linear space. We are looking to confirm if this transformation is beneficial and to what extent it improves the predictions.

5.5.5 Single Versus Combined Output

How well do models perform when predicting with combined metric outputs compared to single metric outputs?

We hypothesise that models trained to predict a single performance metric (e.g., only latency or only throughput) will outperform models trained to predict both metrics simultaneously. The rationale is that a combined-output model has to learn a more complex mapping and potentially deal with conflicting optimisation objectives for the two different metrics. A single-output model can focus all its capacity on learning the patterns for one specific metric. We expect to see lower error rates for the single-output models. We are looking to quantify the performance difference and determine if the convenience of a single, combined model comes at a significant cost in accuracy.

5.6 Results

This section comprehensively presents the empirical results derived from our extensive experimental evaluation of ML models for DDS performance prediction. We detail the predictive accuracy and behaviour of various models under diverse conditions, addressing the hypotheses outlined in Section 5.5. The presentation is structured into two main parts: first, a summary of the initial findings from our previously published work (Peeroo et al. 2024), focusing on LR and RFs in interpolation and extrapolation contexts. Subsequently, we delve into the extended analysis, which explores the impact of network noise, the efficacy of linear versus logarithmic data transformations, and the comparative performance of single versus combined metric output predictions across a broader suite of models including XGB and FFNNs. Our findings illuminate the strengths and weaknesses of each approach, providing critical insights into optimal strategies for the ML modelling of DDS performance metrics.

5.6.1 Initial Results

The initial results are based on the first and published part of the work, which focused on establishing the feasibility of ML for DDS performance prediction using LR and RFs. To recap, one model (LR or RF) is trained for a specific DDS performance metric (e.g., throughput) in interpolation and extrapolation. For each model, the error metrics are taken for the training and testing phases. Therefore, for each DDS performance metric, there are 2 possible models (LR or RF) trained under 2 possible situations (interpolation or extrapolation) with 2 specific error metrics of focus (RMSE and R^2) for the 2 phases (training and testing). This means every model trained on a specific DDS performance metric has 4 variations. Table 5.5 shows all R^2 scores averaged for the model trained for a specific DDS performance metric, while Table 5.6 shows the same for RMSE. We analyse interpolation versus extrapolation for each model by analysing the R^2 and RMSE metrics.

Metric	R2 Train LR Int	R2 Test LR Int	R2 LR Int % Diff	R2 Train LR Ext	R2 Test LR Ext	R2 LR Ext % Diff	R2 Train RF Int	R2 Test RF Int	R2 RF Int % Diff	R2 Train RF Ext	R2 Test RF Ext	R2 RF Ext % Diff
Avg Lost Samples	0.198	0.159	-20	0.083	0.277	234	0.994	0.982	-1	0.996	0.699	-30
Avg Lost Samples (%)	0.251	0.207	-18	0.146	0.237	62	0.984	0.933	-5	0.986	0.907	-8
Avg Received Samples	0.387	0.453	17	0.397	0.028	-93	0.998	0.999	0	0.999	0.780	-22
Avg Received Samples (%)	0.251	0.201	-20	0.24	-0.023	-110	0.986	0.893	-9	0.988	0.908	-8
Avg Sample Rate	0.494	0.534	8	0.431	0.363	-16	0.994	0.969	-3	0.994	0.952	-4
Avg Throughput (Mbps)	0.497	0.536	8	0.511	0.192	-62	0.982	0.94	-4	0.986	0.921	-7
Latency (μ s)	0.083	0.225	171	0.117	0.029	-75	0.723	0.636	-12	0.734	0.188	-74
Total Lost Samples	-0.268	-62.23	-23120	0.009	-0.187	-2178	0.113	-1.211	-1172	0.311	-1.816	-684
Total Received Samples	0.313	0.241	-23	0.275	-0.082	-130	0.984	0.922	-6	0.965	0.427	-56
Total Sample Rate	0.520	0.259	-50	0.434	-1.548	-457	0.958	0.885	-8	0.898	0.711	-21
Total Throughput (Mbps)	0.333	0.370	11	0.522	0.466	-11	0.987	0.939	-5	0.988	0.901	-9

Table 5.5: R2 values during training and testing for LR and RF for interpolation (int) shown in green and extrapolation (ext) shown in blue.

Metric	RMSE Train LR Int	RMSE Test LR Int	RMSE Train LR Ext	RMSE Test LR Ext	RMSE Train RF Int	RMSE Test RF Int	RMSE Train RF Ext	RMSE Test RF Ext
Avg Lost Samples	898,871.4	988,578.1	980,807.9	1,044,115.3	77,933.5	142,382.6	59,083.9	123,808.7
Avg Lost Samples (%)	14.4	13.7	14.5	13.2	1.9	3.2	1.6	4.4
Avg Received Samples	866,112.6	717,621.7	835,644.4	1,082,257.3	28,213.8	31,133.4	25,825.6	261,104.7
Avg Received Samples (%)	14.4	13.8	13.7	15.6	1.8	4.2	1.5	4.4
Avg Sample Rate	4,704.5	4,026.4	4,711.9	6,796.4	202.0	379.2	205.8	1,126.8
Avg Throughput (Mbps)	25.9	26.4	26.2	30.8	1.6	5.3	1.3	8.6
Latency (us)	3,045,001.4	1,454,728.3	2,848,990.0	9,509,031.8	1,139,337.4	771,102.4	916,733.7	7,903,234.4
Total Lost Samples	7,337,539.8	31,204.0	4,131,444.1	2,691,201.6	7,304,408.1	76,398.2	5,656,594.0	183,124.5
Total Received Samples	756,761.8	791,921.4	712,238.7	1,295,118.4	109,964.1	235,794.3	163,318.2	753,747.7
Total Sample Rate	1,839.6	2,018.7	1,745.8	4,918.2	454.0	615.0	651.0	2,206.5
Total Throughput (Mbps)	326.6	361.5	319.2	278.1	23.1	47.3	23.9	149.6

Table 5.6: RMSE values during training and testing for LR and RF for interpolation (int) as shown in green and extrapolation (ext) as shown in blue.

Interpolation on LR

In the context of our performance analysis, it is evident that LR models may not be the most suitable choice for modelling performance metrics. Among the 11 models under consideration, 3 exhibited R^2 scores below the threshold of 0.2, highlighting their inadequate performance. The highest R^2 score recorded in our analysis was 0.54, achieved by the model focusing on the average throughput metric. Another model, specifically designed to capture variations in the average sample rate, demonstrated relatively promising results with an R^2 score of 0.53. Further inspection reveals 3 additional models surpassed the 0.4 R^2 threshold, while 4 models exceeded 0.3. In summary, our comprehensive evaluation indicates that the overall performance of these LR models can be characterised as sub-optimal for the given dataset and metrics.

Extrapolation on LR

The observed performance for extrapolation is notably inferior to that achieved by interpolation methods. Specifically, only 4 of the considered models managed to attain R^2 scores exceeding the threshold of 0.2. Among these models, a mere 2 could surpass a more stringent threshold of 0.3. Notably, the highest-performing model in our analysis was associated with the total throughput metric, achieving an R^2 score of 0.47. Our rigorous evaluation underscores the sub-optimal performance of the models in question, which fails to meet the desired standards.

Metric	R^2 LR Int	R^2 LR Ext	R^2 LR % Diff
Avg Lost Samples	0.159	0.277	74
Avg Lost Samples (%)	0.207	0.237	14
Avg Received Samples	0.453	0.028	-94
Avg Received Samples (%)	0.201	-0.023	-111
Avg Sample Rate	0.534	0.363	-32
Avg Throughput (Mbps)	0.536	0.192	-64
Latency (μ s)	0.225	0.029	-87
Total Lost Samples	-62.23	-0.187	100
Total Received Samples	0.241	-0.082	-134
Total Sample Rate	0.259	-1.548	-698
Total Throughput (Mbps)	0.370	0.466	26

Table 5.7: Interpolation (int) versus extrapolation (ext) for LR focusing on the R^2 metric. Rows with positive percentage differences are highlighted in bold.

Table 5.7 compares R^2 scores between interpolation and extrapolation techniques in LR. The focus here lies on instances where extrapolation exhibits superior performance, resulting in a positive percentage difference.

Metric	RMSE LR Int	RMSE LR Ext	RMSE LR % Diff
Avg Lost Samples	988,578.1	1,044,115.3	6
Avg Lost Samples (%)	13.7	13.2	-4
Avg Received Samples	717,621.7	1,082,257.3	51
Avg Received Samples (%)	13.8	15.6	13
Avg Sample Rate	4,026.4	6,796.4	69
Avg Throughput (Mbps)	26.4	30.8	17
Latency (us)	1,454,728.3	9,509,031.8	554
Total Lost Samples	31,204.0	2,691,201.6	8525
Total Received Samples	791,921.4	1,295,118.4	64
Total Sample Rate	2,018.7	4,918.2	144
Total Throughput (Mbps)	361.5	278.1	-23

Table 5.8: Interpolation (int) versus extrapolation (ext) for LR focusing on the RMSE metric. Negative values are highlighted in bold.

In 4 specific cases, extrapolation outperforms interpolation: Average Lost Samples, Average Lost Samples (%), Total Lost Samples, and Total Throughput. Regarding the Lost Samples category, it's worth noting that a significant portion of the data consists of zeroes, making extrapolation advantageous. In this scenario, the model predicts zero values due to the high frequency of zeroes in the data. This is evident in the R^2 scores, where interpolation achieves 0.16, while extrapolation notably improves to 0.28, as observed in Average Lost Samples.

An intriguing case arises with Total Throughput. Interpolation yields an R^2 score of 0.37, while extrapolation achieves an impressive 0.47, marking a noteworthy 26% improvement. This suggests that LR has the potential for extrapolation, albeit with room for improvement in R^2 performance.

In examining the RMSE, we find that in the case of Total Throughput, the RMSE decreases from 362 to 278, implying a 23% reduction. This reduction in RMSE reinforces that LR holds promise for extrapolation but underscores the need for continued improvement. This case demonstrates that LR can yield better results in extrapolation than interpolation, particularly for metrics like Total Throughput.

Interpolation on RFs

In the evaluation of the 11 models, 1 model fell within the 0.6 to 0.7 range in terms of performance, specifically in relation to latency. 2 models scored between 0.8 and 0.9, namely the average received samples percentage and the total sample rate. Notably, 7 models of the 11 demonstrated excellent performance, each scoring higher than 0.9.

The model with the lowest performance registered a score of -1.2, associated with the total

lost samples. Conversely, the best-performing model achieved a near-perfect score of 0.999, specifically in the category of predicting average received samples.

Overall, an impressive 9 of the 11 models displayed incredibly good performance in terms of the R^2 metric. Only 1 model was considered mediocre (latency with a score of 0.636), and the worst-performing model was significantly poor (total lost samples with a score of -1.211). The 9 models that exhibited superior performance can be seen in Table 5.9, which compares interpolation and extrapolation for RFs focusing on the R^2 metric.

Metric	R^2 RF Int	R^2 RF Ext	R^2 RF % Diff
Avg Lost Samples	0.982	0.699	-29
Avg Lost Samples (%)	0.933	0.907	-3
Avg Received Samples	0.999	0.780	-22
Avg Received Samples (%)	0.893	0.908	2
Avg Sample Rate	0.969	0.952	-2
Avg Throughput (Mbps)	0.940	0.921	-2
Latency (μ s)	0.636	0.188	-70
Total Lost Samples	-1.211	-1.816	-50
Total Received Samples	0.922	0.427	-54
Total Sample Rate	0.885	0.711	-20
Total Throughput (Mbps)	0.939	0.901	-4

Table 5.9: Interpolation (int) versus extrapolation (ext) for RFs focusing on the R^2 metric.

The RMSE is challenging to interpret in isolation, requiring a comparison with the average value of the metric for meaningful insights. To address this, the RMSE has been divided by the mean for all metrics under consideration, producing the Relative Root Mean Square Error (RRMSE).

The lowest RRMSE observed was 4% for the average received samples percentage, indicating a high degree of accuracy for this model. On the other end of the spectrum, the worst relative error was found in latency, at a significant 64%. 2 models had an RRMSE of less than 10%, specifically average received samples and average received samples percentage. Both metrics are closely related, as they represent the same underlying data but in different forms and we therefore expect them to have similar RRMSEs. 2 models fell within the 10 to 20% range, namely average sample rate and average throughput. 3 models exhibited RRMSEs between 20 to 30%, including total lost samples, total sample rate, and total throughput. One model was in the 30 to 40% range, specifically total received samples. Finally, 2 models had their RRMSEs between 50 to 60%, which were average lost samples and average lost samples percentage.

The most outstanding model in this analysis was the average received samples model, boasting an R^2 of 0.999 and a remarkably low RRMSE of 4%. This reflects its exceptional accuracy

and precision. Additionally, several other models demonstrated very good overall performance. The average received samples percentage model, with an R^2 of 0.89 and an RRMSE of 4%, performed almost as well as the top model, albeit with a slightly lower R^2 score. The average sample rate model showed a strong correlation with an R^2 of 0.95, though its RRMSE was higher at 33%. Similarly, the average throughput model recorded an R^2 of 0.94 and an RRMSE of 13%, indicating a high level of accuracy with a moderate error relative to the mean. Finally, the total throughput model also presented a strong predictive capability with an R^2 of 0.94, but a somewhat higher RRMSE of 21%, reflecting a reasonable level of error in relation to the mean.

A comparison of RF with LR for interpolation can be observed in Table 5.5. Comparing the columns labelled “ R^2 Test LR Int” and “ R^2 Test RF Int”, it is evident that RFs outperform LR in all cases. This indicates that RFs are a more suitable choice for modelling DDS performance metrics, particularly in interpolation scenarios.

Extrapolation on RFs

5 models scored an R^2 between 0.9 and 1 for their performance in extrapolation, demonstrating high efficacy, while 2 models were in the 0.7 to 0.8 range. The average received samples model was noted at 0.78, and the total received samples model was considerably lower at 0.43. Latency, with a value of 0.19, was identified as particularly poor. Scatter plots shown in Figure 5.4 indicated a trend where poor predictions often significantly underestimated actual values. Notably, 5 out of the 11 models excelled in extrapolation, a critical aspect for predictive analysis. These models included average lost samples percentage, average received samples percentage, average sample rate, average throughput, and total throughput, highlighting that percentage-based metrics are generally more reliable for extrapolation, especially in throughput-related data where percentage values are more insightful than absolute figures. Comparing the R^2 values between RF and LR for extrapolation can be observed in Table 5.5 when focusing on the columns labelled “ R^2 Test LR Ext” and “ R^2 Test RF Ext”. Just like in interpolation, RFs outperform LR in all cases, indicating that RFs are a more suitable choice for modelling DDS performance metrics in both interpolation and extrapolation scenarios.

Focusing on RRMSE, the average received samples percentage model emerged as the best, with an RRMSE of 5%. The average throughput model followed with an RRMSE of 22%, and the average sample rate had an RRMSE of 33%. 2 models each had an RRMSE of 47% and 60-70%, respectively, while the total sample rate’s RRMSE was disappointingly high at

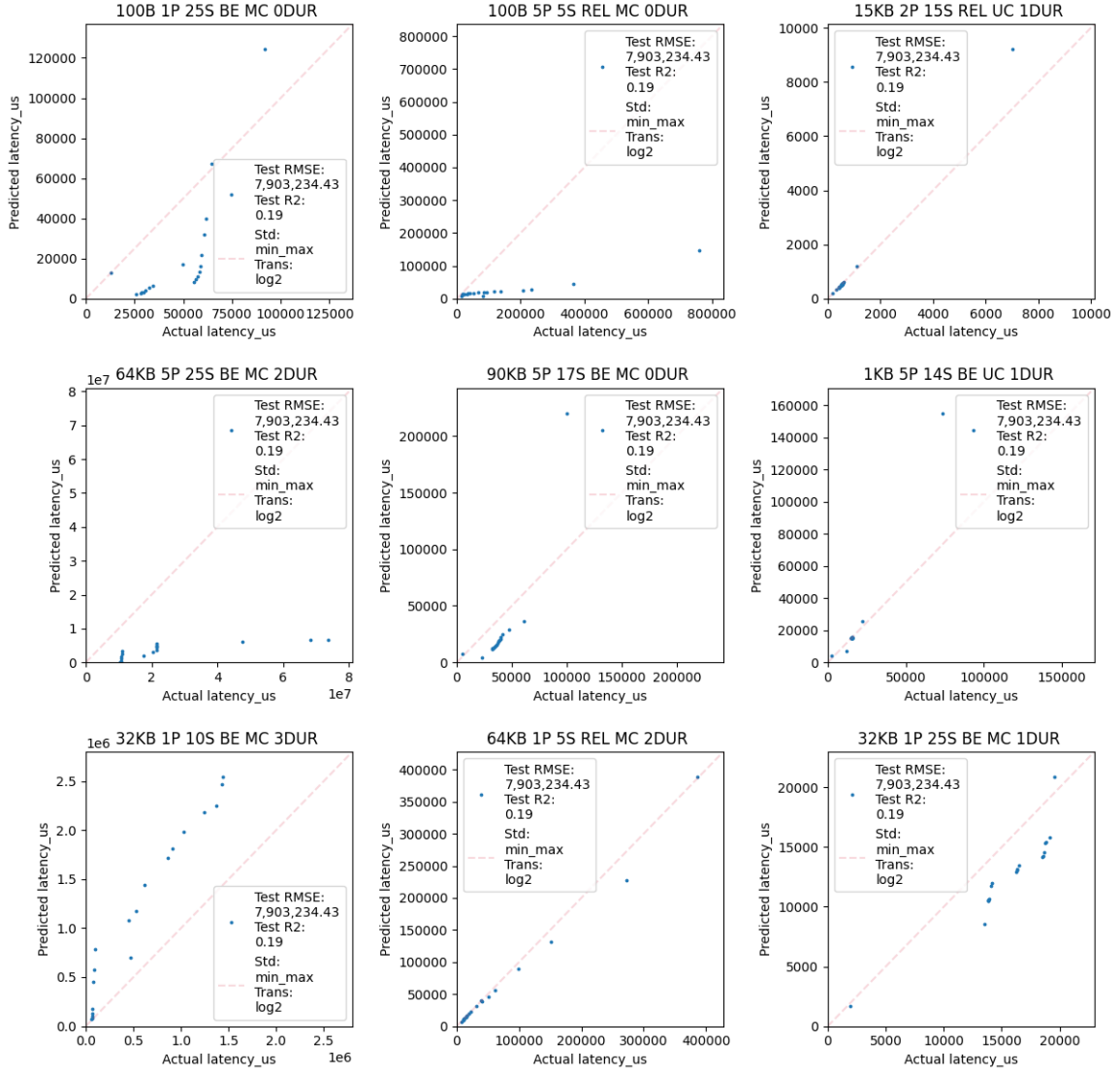


Figure 5.4: Scatter plots of predicted values on y-axis and actual values on x-axis for 9 randomly picked tests using the RFs model to extrapolate for latency.

94%, and the total received samples even higher at 108 %. Latency fared the worst with an RRMSE of 654%, due to cases where predictions were significantly off, resulting in high errors that skewed the average. To further understand this, an analysis of relative errors is necessary.

Moreover, the best performing model in terms of R^2 was the average received samples percentage, with an R^2 of 0.91 and an RRMSE of 5%. Other models showing commendable performance included the average sample rate with an R^2 of 0.95 and RRMSE of 33%, and the average throughput with an R^2 of 0.92 and RRMSE of 22%. In stark contrast, latency was the worst-performing model with an R^2 of 0.19 and an RRMSE of 654%, and the total lost samples model also fared poorly with an R^2 of -1.82 and an RRMSE of 69%.

Metric	RMSE RF Int	RMSE RF Ext	RMSE RF % Diff
Avg Lost Samples	142,382.6	123,808.7	-13
Avg Lost Samples (%)	3.2	4.4	38
Avg Received Samples	31,133.4	261,104.7	739
Avg Received Samples (%)	4.2	4.4	5
Avg Sample Rate	379.2	1,126.8	197
Avg Throughput (Mbps)	5.3	8.6	62
Latency (μ s)	771,102.4	7,903,234.4	925
Total Lost Samples	76,398.2	183,124.5	140
Total Received Samples	235,794.3	753,747.7	220
Total Sample Rate	615.0	2,206.5	259
Total Throughput (Mbps)	47.3	149.6	216

Table 5.10: Interpolation (int) versus extrapolation (ext) for RFs focusing on the RMSE metric. Negative values are highlighted in bold indicating improvement with extrapolation.

Summary

In summary, the comprehensive evaluation of 11 RF models revealed a diverse range of performances across different metrics. The analysis highlighted a clear distinction in model capabilities, particularly in terms of R^2 scores and RRMSE. Models like the average received samples and its percentage variations demonstrated exceptional accuracy, with high R^2 scores and low RRMSEs, indicating their robustness in both interpolation and extrapolation scenarios. Conversely, models such as latency and total lost samples performed poorly, characterised by low R^2 values and high RRMSEs, suggesting significant room for improvement in these areas.

The findings also underline the importance of considering both the R^2 metric and RRMSE for a holistic understanding of model performance. While some models showed high R^2 scores, their relative errors were significantly higher, emphasising the need for a balanced approach in model evaluation. The analysis of scatter plots further enriched these insights, revealing trends in prediction accuracy and the tendency of some models to underestimate values in certain scenarios.

Overall, this evaluation of RF models underscores the effectiveness of certain models in handling complex data predictions, especially in cases where percentage values are more relevant than absolute numbers. However, it also points to the challenges faced by models in accurately predicting metrics with a high variance or a significant number of zero values. These insights are invaluable for guiding future improvements and optimisations in model development.

When comparing the results of RFs with LR for interpolation and extrapolation, we observe that RFs consistently outperform LR across all metrics. Thus, RFs are a more suitable choice

for modelling DDS performance metrics in both interpolation and extrapolation scenarios. To conclude the initial results, we can state that RF models are a good choice for modelling DDS performance metrics and thus we mark the successful application of ML to DDS performance prediction.

The rest of this chapter presents results for the extended work, which includes additional models and analyses. To recap, the extended work differs in the following ways:

- 2 new models: XGB and FFNNs
- Focus only on Latency and Throughput DDS metrics
- Split the dataset differently: train, test, validation, and out-of-sample
- One new testbed: 5Pi
- 3 noise levels per testbed: no noise, noise, and extra noise
- Focus on the impact of data transformations (linear versus logarithmic)
- Focus on the impact of predicting multiple metrics at once

We start off by building on top of the comparison between interpolation and extrapolation in Section 5.6.2, which was done for LR and RFs.

5.6.2 Interpolation Versus Extrapolation

Within the extended work, we investigate to see if there are cases where extrapolation performs well and we quantify this by defining an R^2 value above 0.8 as performing well. To recap, the extended work introduced many new dimensions resulting in a large number of models trained and evaluated. To carry out the analysis comparing interpolation performance to extrapolation performance we did the following. For each metric (latency, throughput or both combined), model (LR, RF, XGB, or FFNN), testbed (3pi or 5pi), and BW cap level (no cap, moderate cap, or high cap), we iterate and gather the R^2 during validation and out-of-sample testing. Since there are ten runs per model we can calculate confidence intervals with a significance level of 95% for the R^2 values. We break down the results into two sections. In the first subsection, we do an initial analysis of the results and look for cases where extrapolation works well. We then look at the results in more detail, where we look for cases where extrapolation outperforms interpolation.

Are there cases where extrapolation works well?

There were 17 cases indicating that extrapolation works well, with an R^2 value greater than or equal to 0.8 and these cases are listed in Table 5.11.

Metric	Model	Output Transformation	Dataset
Throughput	RF	log1p	3Pi No
Throughput	RF	log1p	5Pi 24 Mbps
Latency	RF	log1p	3Pi 8 Mbps
Throughput	XGB	log1p	3Pi No
Latency	XGB	log1p	3Pi 8 Mbps
Latency	RF	log1p	3Pi 6 Mbps
Throughput	XGB	log1p	5Pi 24 Mbps
Combined	RF	log1p	5Pi 24 Mbps
Latency	XGB	log1p	3Pi 6 Mbps
Throughput	FFNN	log1p	5Pi 24 Mbps
Combined	XGB	log1p	5Pi 24 Mbps
Combined	RF	log1p	3Pi 8 Mbps
Combined	RF	log1p	3Pi 6 Mbps
Throughput	XGB	none	5Pi 24 Mbps
Latency	RF	log1p	5Pi 24 Mbps
Throughput	RF	none	5Pi 24 Mbps
Throughput	FFNN	none	5Pi 24 Mbps

Table 5.11: Specific Model Configurations Where Extrapolation R^2 is Greater Than or Equal to 0.8

Majority of cases where extrapolation works well are for throughput metrics, with 8 out of 17 cases. Latency metrics account for 5 cases, and the remaining 4 cases are for combined metrics, which include both latency and throughput combined as the predicted output. In terms of which models appeared most, RFs are the most common seen in 9 out of 17 cases, followed by XGB models in 6 instances, while FFNNs accounted for the remaining 2 cases. The logarithmic output transformation (**log1p**) was predominantly used in 14 out of 17 cases, while the linear transformation was employed in the remaining 3 instances indicating a preference for logarithmic scaling in scenarios where extrapolation performs well. Finally, the 5Pi 24 Mbps dataset was seen most (9 of 17 cases) where extrapolation performed well. This indicates that while extrapolation generally presents challenges, it can yield effective results under specific conditions. It is generally not expected for tree-based models, such as RFs and XGB, to excel at extrapolation, making these results surprising. Further validation of these findings is necessary to understand the underlying reasons for this atypical behavior.

Are there cases where extrapolation outperforms interpolation?

To identify if extrapolation outperforms interpolation, we compare the validation mean upper bound with the out-of-sample mean lower bound and look for cases where the lower bound of extrapolation is greater than the upper bound of interpolation. These cases are shown in Table 5.12. It should be noted that the differences between interpolation and extrapolation R^2 are

Metric	Model	Input Transformation	Output Transformation	Dataset
Combined	RF	none	log1p	5Pi 24 Mbps
Combined	XGB	minmax	log1p	5Pi 24 Mbps
Latency	XGB	minmax	log1p	3Pi 8 Mbps
Throughput	RF	none	log1p	5Pi 24 Mbps

Table 5.12: Cases where Extrapolation Outperforms Interpolation

not very significant, ranging from 0.0064 to 0.0336, and should be interpreted with caution. These differences are not statistically significant, and we have not performed tests to check for statistical significance because there are only 4 cases and the differences are so insignificant. However, we still objectively report these cases as avenues for further investigation.

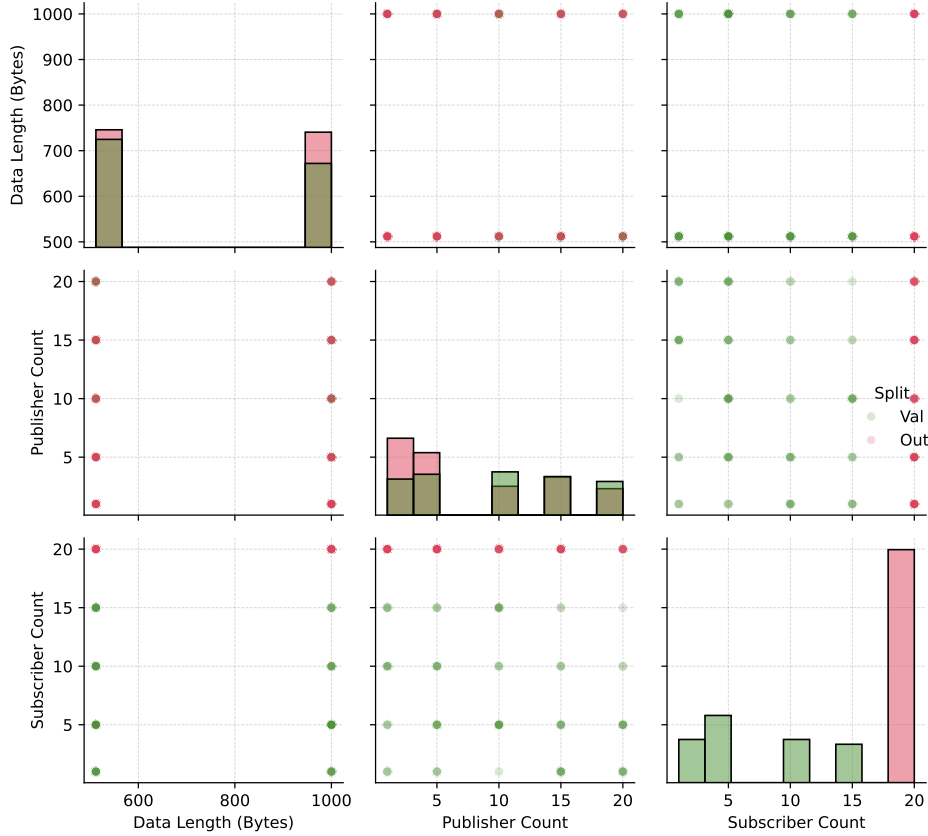


Figure 5.5: Input Distributions for Validation and Out-of-Sample Datasets for 5Pi 24 Mbps Dataset

The significantly positive results for extrapolation are unexpected, especially for tree-based models like RFs and XGB, which are generally not known for their extrapolation capabilities. We need to validate these results and start by analysing the input distributions of the data to see how different the range of input values are for validation and out-of-distribution for the 5Pi 24 Mbps dataset explicitly (because of its appearance in 3 of the 4 cases). Analysis of the input distributions in Figure 5.5 reveals an overlap in the “Data Length” and “Publisher Count” features between the validation and out-of-sample datasets. It should be noted that

the 3 squares in the lower left corner of the figure are analogous to the 3 squares in the upper right corner. Conversely, a clear separation was observed for the “Subscriber Count” feature, indicating that the models were trained on a different distribution of “Subscriber Count” values compared to the out-of-sample dataset. This suggests that the models had to extrapolate for the “Subscriber Count” feature when making predictions on the out-of-sample dataset. This observation could potentially explain why extrapolation outperformed interpolation in these specific cases.

5.6.3 Network Noise

Our published work can be criticised for not being realistic enough (despite the existence of real-life scenarios using isolated LANs without noise e.g. robots), because we collected performance data in an isolated network environment without any network noise. We would like to investigate the impact of network noise on the models by adding a BW cap to the network. We introduce 3 BW cap levels: no cap, low cap, and high cap, as mentioned in Section 3.4. Therefore, for each testbed, for each DDS metric (latency and throughput), and for each model we use statistical analysis to compare the performance across the 3 cap levels. We first determine normality using the Shapiro-Wilk (W) Test (SHAPIRO & WILK 1965). If all 3 BW cap levels distributions are normally distributed, we proceed with comparison using Analysis of Variance (ANOVA) (Sthle & Wold 1989). Otherwise, we use the Kruskal-Wallis (H) Test (Kruskal & Wallis 1952). This will tell us if the 3 groups are statistically significantly different. We then use Tukey’s Honest Significant Difference (HSD) Test (Tukey 1949) (if normal) or Dunn’s Test (Dinno 2015) (if not normal) to determine the ranking of the 3 groups while also visually comparing the eCDF of the 3 groups.

The analysis begins by presenting the initial data exploration, which includes descriptive statistics for mean R^2 values across cap levels. This is followed by the statistical significance of BW cap impact, which includes statistical tests (ANOVA and H) to assess the significance of R^2 variations across cap levels. The performance extremes section that follows presents an analysis of which BW cap levels most frequently correspond to the lowest and highest mean/median R^2 values. Finally, the analysis by testbed, metric, and model sections provide a breakdown of the results by testbed, metric, and model respectively.

Overview

Descriptive statistics for mean R^2 values across cap levels reveal the impact of BW cap levels on model performance as shown in Table 5.13. The mean R^2 for the “No” cap is -0.24, while “Low” and “High” caps show substantially lower averages of -43.17 and -13.36, respectively. This challenges the assumption that less noise (as in “High” cap) always leads to better R^2 scores. Standard deviations for “Low” (295.889) and “High” (39.0982) caps are considerably higher than for “No” cap (1.358), indicating reduced reliability under capped conditions. Extremely low minimum R^2 values for “Low” (-2050.35) and “High” (-204.10) caps highlight critical failure points for certain model configurations under capped conditions. However, despite these major differences, the 50%, 75%, and maximum R^2 values are in fact higher for the “Low” cap compared to both “No” and “High” caps indicating that some models perform better with a “Low” cap and this warrants further investigation.

Statistic	No Cap	Low Cap	High Cap
mean	-0.242	-43.167	-13.357
std	1.358	295.889	39.098
min	-6.311	-2050.350	-204.103
25%	-0.032	-0.072	-1.598
50%	0.211	0.445	0.255
75%	0.414	0.672	0.520
max	0.891	0.984	0.906

Table 5.13: Descriptive Statistics for Mean R^2 Values Across BW Cap Levels

Statistical Analysis

In 46 out of 48 cases, both ANOVA and H tests showed statistically significant differences in R^2 values across “No”, “Low”, and “High” cap levels. This confirms that BW cap levels measurably impact model performance. We conducted pairwise comparisons to identify specific differences between cap levels using HSD (for normally distributed data) and Dunn’s Test (for non-normally distributed data). Most pairwise comparisons reveal statistically significant differences between the “No” cap level and both “Low” and “High” cap levels. “No” cap consistently yields higher R^2 values, reinforcing that uncapped environments generally lead to superior model performance. The relationship between “Low” and “High” caps is nuanced, with varying directions of differences depending on the specific model and transformation. This suggests that the impact of cap levels within capped scenarios is not consistently predictable, requiring further analysis.

Performance Analysis

Statistic	No Count	Low Count	High Count
Lowest Mean	21	10	17
Highest Mean	17	20	11
Lowest Median	22	11	15
Highest Median	15	20	13
Total	75	61	56

Table 5.14: Counts of Lowest and Highest Mean/Median R^2 by BW Cap Level

This section delves into the extremes of model performance across different BW cap levels. This is further analysed by breaking down the results by testbed, metric, and model in the subsequent sections. The worst and best performing scenarios are captured by the lowest/highest mean and median R^2 values for each cap level. The number of cases where each cap level corresponds to the lowest and highest mean/median R^2 values is summarised in Table 5.14. Each value corresponds to the number of models (out of 48) where that cap level produced the respective extreme R^2 value in comparison to the other cap levels. The “No” cap, while capable of the best performance, frequently leads to the lowest mean and median R^2 scores (21 and 22 times, respectively), indicating a risk of poor performance in uncapped environments. The “Low” BW cap levels surprisingly often yields the highest mean and median R^2 scores (20 times each), suggesting that moderate noise can be beneficial, possibly acting as a regulariser. Despite expectations, the “High” BW cap levels is least likely to produce the highest R^2 scores, implying that simply minimising noise isn’t always the best strategy for maximising R^2 performance.

Testbed	Mean	No Count	Low Count	High Count
3Pi	Lowest	5	6	13
	Highest	13	9	2
5Pi	Lowest	16	4	4
	Highest	4	11	9

Table 5.15: Lowest and Highest Mean R^2 Counts by Testbed and BW Cap Level

Testbed Analysis A critical aspect of our analysis is understanding how system scale impacts model predictability. By comparing the performance across the 3Pi and 5Pi testbeds, we can evaluate the models’ robustness to increased network complexity. Therefore, we further investigate the performance extremes by breaking them down by testbed as shown in Table 5.15.

The impact of BW cap levels on R^2 scores varies significantly between the 2 testbeds. On the 3Pi testbed, the “No Cap” environment most frequently yielded the highest mean R^2 (13 times), but also demonstrated high variability, appearing as the lowest-performing scenario 5 times. The “High” cap was most often detrimental, leading to the lowest mean R^2 in 13 instances.

In contrast, on the more complex 5Pi testbed, the “No Cap” scenario was overwhelmingly the worst-performing, resulting in the lowest mean R^2 in 16 out of 24 cases. This suggests that the unconstrained network traffic from a larger number of nodes leads to saturation and unpredictable behaviour, making performance harder to model. Interestingly, the “Low” cap most frequently achieved the highest mean R^2 (11 times), indicating that a moderate level of noise may act as a regulariser, improving model generalisation in a more complex environment.

We hypothesise that the increased number of nodes in the 5Pi testbed introduces greater network contention and more complex communication patterns. This heightened complexity makes the system’s performance inherently less predictable, especially in uncapped scenarios. The results suggest that while some models can adapt, the increase in scale presents a significant challenge to performance prediction. This highlights that the optimal noise mitigation strategy is highly dependent on the specific system scale and architecture, and a “one-size-fits-all” approach is not suitable.

Metric	Mean	No Count	Low Count	High Count
Combined	Lowest	6	3	7
	Highest	5	9	2
Latency	Lowest	4	4	8
	Highest	7	8	1
Throughput	Lowest	11	3	2
	Highest	5	3	8

Table 5.16: Lowest and Highest Mean R^2 Counts by Metric and BW Cap Level

Metric Analysis The predictive accuracy of our models is not only influenced by the testbed but also by the intrinsic nature of the performance metric being evaluated. Latency and throughput exhibit fundamentally different behaviours under network constraints, which in turn affects how easily they can be modelled.

Table 5.16 breaks down the performance extremes by metric. For latency prediction, a clear pattern emerges where moderate noise levels are most beneficial. The “Low” cap most frequently yields the highest mean R^2 scores (8 times), while the “High” cap most often results in the lowest scores (8 times). We hypothesise that this is due to the nature of latency data,

which often contains a long tail of high-value outliers. A “Low” cap may regularise the data by constraining these extreme spikes, creating a more predictable distribution. Conversely, a “High” cap might be insufficient to control these outliers, leading to noisier data that is harder to model effectively.

In stark contrast, throughput prediction appears to benefit from more constrained environments. The “No Cap” scenario most frequently leads to the lowest predictive performance (11 times), whereas the “High” cap most often produces the best results (8 times). This suggests that uncapped throughput, while potentially higher, is also more volatile and thus less predictable. A “High” cap likely stabilises the data rate, removing erratic fluctuations and creating a more consistent pattern for the models to learn.

Mean	Model	No Count	Low Count	High Count
Lowest	LR	6	2	4
	RF	4	1	7
	XGB	4	2	6
	FFNN	7	5	0
Highest	LR	6	4	2
	RF	3	7	2
	XGB	4	5	3
	FFNN	4	4	4

Table 5.17: Lowest and Highest Mean R^2 Counts by Model and BW Cap Level

Model Analysis The response to network noise is not uniform across the different ML models, with performance being highly dependent on the model’s underlying architecture. Our results, as shown in Table 5.17, reveal a clear distinction between the behaviour of ensemble tree-based models and that of the other models.

RF and XGB, both being ensemble methods, demonstrate a similar and notable trend. They generally perform best under moderate noise, with the “Low” cap most frequently yielding the highest R^2 scores. This suggests that their ensemble structure, which averages predictions from many individual trees, provides inherent robustness against some level of noise. The moderate noise from the “Low” cap may even act as a beneficial regulariser, preventing the models from overfitting to the specific conditions of the uncapped training data. Conversely, they are most negatively impacted by the “High” cap, which most often leads to their lowest performance, indicating that this level of noise is sufficient to degrade predictive accuracy without providing the same regularising benefit.

In contrast, the FFNN and LR models show greater sensitivity to uncapped environments. FFNNs most frequently record their lowest R^2 scores with the “No” cap, likely due to their

sensitivity to the wide variance and potential outliers present in unconstrained network conditions. LR is highly variable in the “No” cap scenario, frequently producing both the highest and lowest R^2 scores, which underscores its instability when faced with complex, non-linear data patterns.

This analysis demonstrates that the choice of ML model significantly influences how BW cap levels affect R^2 performance. Ensemble models like RF and XGB appear more adept at handling, and even benefiting from, certain noise characteristics, whereas FFNNs and LR are more susceptible to the volatility of unconstrained environments.

5.6.4 Linear Versus Log Space

The choice between linear and logarithmic transformations for modelling performance metrics is crucial, as it directly influences the interpretability and predictive accuracy of the models. A logarithmic transformation compresses larger values, thereby reducing the overall range of the data. Conversely, a linear representation maintains the original scale, which can result in a wide variance, particularly for extreme values such as worst-case latency. For values that are not considered worst-case or extremely high, a logarithmic transformation is expected to yield superior predictive performance. This is because logarithmic transformations can normalise skewed distributions and reduce the influence of outliers, making the data more amenable to modelling. In summary, it is hypothesised that linear transformations will result in higher R^2 values for worst-case latency percentiles, while logarithmic transformations will demonstrate superior performance for other latency percentiles. This implies that linear models are expected to outperform logarithmic models for worst-case latency percentiles, and vice versa for other percentiles.

To rigorously evaluate these hypotheses, a comprehensive statistical methodology is employed. Initially, the W test is used to assess the normality of the data distributions for both linear and logarithmically transformed results. Depending on the outcome of the normality test, either Welch’s t-test (WELCH 1947) (for normally distributed data) or the Mann-Whitney U (U) test (Mann & Whitney 1947) (for non-normally distributed data) is applied to determine statistical significance. Additionally, the KS test is used to compare the cumulative distributions. The determination of statistical significance is based on the following criteria:

- Both distributions are normal, and both Welch’s t-test and KS test indicate significance.
- Both distributions are non-normal, and both the U test and KS test indicate significance.
- One distribution is normal and the other is not, and both the U test and KS test indicate

significance.

This systematic approach ensures a statistically sound conclusion regarding whether linear or logarithmic transformations yield superior performance in specific contexts.

To further refine the analysis, percentiles are categorised into distinct scopes:

- **Worst-Case Percentiles:** Comprising the 95th, 97th, 98th, 99th, and 100th percentiles.
- **All Percentiles:** Encompassing the entire range from 0 to 100.
- **Middle Percentiles:** Specifically, the 40th, 50th, and 60th percentiles.
- **Normal-Case Percentiles:** Including all percentiles excluding the worst-case percentiles (0, 1, 2, 3, 5, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90).

These scopes are designed to isolate different performance characteristics, allowing for a more targeted analysis of how linear and logarithmic transformations perform under varying conditions. The aim of this section is to validate the initial hypothesis by examining the performance of linear versus logarithmic transformations across these defined percentile scopes. This section is broken into two parts: latency results and throughput results. Each part is further broken down by the different percentile scopes, starting with normal and worst-case percentiles followed by overall.

Latency

Cases where both linear and logarithmic transformation results yielded negative values were excluded from the analysis. Consequently, the focus of this investigation is exclusively on positive outcomes. It is hypothesised that linear transformations will demonstrate superior performance for worst-case percentiles. Conversely, logarithmic transformations are anticipated to perform better for normal-case percentiles. This expectation is predicated on the understanding that worst-case percentiles typically encompass larger values, which inherently exhibit greater variance. This section presents the results for normal and worst-case percentiles first, followed by the overall results.

Normal and Worst-Case Percentiles Analysis Tables 5.18, 5.19, 5.20, and 5.21 summarise the results of the linear versus logarithmic transformation analysis for latency across different models and datasets. From these results, we notice that RF models consistently achieved high R^2 scores, notably reaching 0.9962 for the 3Pi 6 Mbps worst-case percentiles as shown in Table 5.20. This strong performance is attributed to their ensemble averaging, which effectively reduces variance and is well-suited for modeling non-linear latency patterns. Both

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi 6 Mbps	Normal	0.0683	-0.0466	0.1223	-0.0335	linear
	Worst	0.1751	-0.0342	0.1733	-0.0345	linear
3Pi 8 Mbps	Normal	-0.0622	-0.1858	0.0641	-0.0424	linear
	Worst	0.1124	-0.0431	0.1131	-0.0435	linear
5Pi 24 Mbps	Normal	0.1808	-0.0640	0.1928	-0.0678	linear
	Worst	0.3374	-0.1559	0.3379	-0.1550	linear
5Pi 71 Mbps	Normal	-17.3118	-0.1342	0.0489	-0.0340	–
	Worst	0.2469	-0.1020	0.2432	-0.0899	linear
5Pi No	Normal	-34.6758	0.1070	-6.1213	0.0720	–
	Worst	0.1685	-0.0020	0.1459	-0.0131	linear

Table 5.18: Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for LR Models. Bigger value indicates the transformation that yields a higher R^2 score.

RF and XGB models demonstrated strong performance on the 3Pi 6 Mbps and 3Pi 8 Mbps datasets. For the 3Pi 6 Mbps dataset, the linear distribution was heavy-tailed and positively skewed, while the log-transformed data exhibited a normal distribution with very high kurtosis. RF models excel at identifying and modeling such complex patterns. XGB’s iterative error correction mechanism further aids in discovering intricate relationships, even within smaller datasets.

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
5Pi 71 Mbps	Normal	-8.8486	0.0664	0.0432	0.0468	–
	Worst	0.0283	-14.6848	0.0292	-11.0498	linear

Table 5.19: Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for FFNN Models. Bigger value indicates the transformation that yields a higher R^2 score.

Conversely, LR models performed poorly, particularly for the 5Pi datasets under normal-case percentiles conditions, yielding highly negative R^2 values (e.g., -17.3118 for 71 Mbps and -34.6758 for no BW throttle). This sub-optimal performance is likely due to LR’s inability to effectively model the inherent non-linear behaviors present in these datasets. However, LR showed improved performance for the 5Pi 24 Mbps dataset for worst-case percentiles. This suggests the presence of simpler, more linear patterns within this specific dataset that LR could effectively capture.

FFNNs generally exhibited the poorest performance. This could be attributed to overfitting or insufficient hyperparameter tuning, despite a 5-minute allocation for hyperparameter optimisation. A potential reason for this limitation is the constrained learning rate of 0.1, which may be inadequate for effectively modeling the larger latency values observed in the data.

The 3Pi 6 Mbps and 3Pi 8 Mbps datasets proved to be more predictable, particularly for RF

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi 6 Mbps	Normal	0.9119	0.6771	0.9236	0.7731	linear
	Worst	0.9962	0.9872	0.9965	0.9898	linear
3Pi 8 Mbps	Normal	-0.5406	0.1900	0.8853	0.4043	–
	Worst	0.8978	0.6366	0.8899	0.6307	linear
5Pi 71 Mbps	Normal	-65.9915	0.2756	0.1940	0.1949	–
	Worst	0.7267	0.6403	0.7267	0.6680	linear

Table 5.20: Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for RF Models. Bigger value indicates the transformation that yields a higher R^2 score.

and XGB models. In contrast, the 5Pi 71 Mbps and 5Pi No Noise datasets presented greater challenges to predictability. Comparing normal-case percentiles and worst-case percentiles, RF and XGB generally outperformed other models for normal-case percentiles, although results were mixed, with RF occasionally yielding highly negative R^2 values (e.g., -65.9915 for 5Pi 71 Mbps). Conversely, worst-case percentiles consistently resulted in higher R^2 scores across all models, with RF achieving an R^2 of 0.9962 for the 3Pi 6 Mbps dataset. This indicates that worst-case percentiles are inherently more predictable. RF and XGB consistently demonstrated superior performance due to their inherent robustness to outliers.

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi 6 Mbps	Normal	-0.6647	0.5736	0.8296	0.7217	–
	Worst	0.9444	0.8604	0.9468	0.8895	linear
3Pi 8 Mbps	Normal	-4.1495	-0.6094	0.8107	0.1501	–
	Worst	0.9150	0.3005	0.9200	0.3346	linear
5Pi 24 Mbps	Normal	0.5465	0.2667	0.5551	0.3359	linear
	Worst	0.7232	0.6660	0.7139	0.6433	linear

Table 5.21: Latency R^2 Comparison: Linear vs. Logarithmic Transformations for Normal and Worst-Case Percentiles for XGB Models. Bigger value indicates the transformation that yields a higher R^2 score.

Anomalous results were frequently observed for normal-case percentiles, particularly for FFNNs and RF models. This is likely attributable to high variance or pronounced non-linear patterns prevalent in high-BW datasets. For instance, FFNNs with worst-case percentiles for the 5Pi 71 Mbps dataset yielded a significantly negative R^2 of -14.6848. This degradation in performance is possibly exacerbated by the log transformation, which can amplify the impact of a poor model fit. Such outcomes suggest that the network architecture or hyperparameters employed may be unsuitable for these specific conditions.

Table 5.22 summarises the cases where logarithmic transformations outperformed linear transformations for worst-case percentiles. There are only 2 models that exhibit this behaviour: RF and XGB. This was also observed in the 3Pi No dataset, which is characterised by a heavy-

Model	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
RF	Normal	-13.6379	0.3029	-0.0457	0.2720	–
	Worst	0.3336	0.4620	0.3209	0.4532	log
XGB	Normal	0.8345	0.7120	0.8989	0.7122	linear
	Worst	0.1769	0.6705	0.2403	0.7644	log

Table 5.22: Cases where Logarithmic Transformation Outperforms Linear Transformation for Worst-Case Percentiles for Latency (all cases were 3Pi No Cap)

tailed distribution. Modelling in linear space for the RF model resulted in a highly negative R^2 of -13.6379, indicating that the linear transformation was not effective for this dataset. Conversely, the log transformation yielded a mean R^2 of 0.3029, suggesting that it was more effective in capturing the underlying patterns of the data. In both cases, normal and worst-case, modelling in the log space yielded better results than in linear space. More importantly, the models that were trained in log space performed better on worst-case percentiles than on normal-case percentiles indicated by the difference in the mean (0.3029 for normal-case percentiles versus 0.4620 for worst-case percentiles) and the difference in the median (0.2720 for normal-case percentiles versus 0.4532 for worst-case percentiles).

For XGB models, linear transformations surprisingly surpassed log transformations with normal-case percentiles, a difference that was statistically significant. This trend completely reversed with worst-case percentiles, where log transformations significantly outperformed linear transformations, also with statistical significance and a much larger margin than observed with RF. This heightened sensitivity of XGB to data transformations, compared to RF, is likely due to its iterative optimisation process. RF, by averaging across multiple decision trees, tends to be less sensitive to individual data transformations.

Linear transformations are more likely to outperform logarithmic transformations for worst-case percentiles, with the notable exception of the 3Pi No dataset when using RF and XGB models. RF and XGB consistently excel when linear transformations yield superior results for worst-case percentiles, particularly evident with the 3Pi 6 Mbps and 3Pi 8 Mbps datasets. The 3Pi 8 Mbps dataset exhibits noticeable peaks, indicating low variation, which simplifies prediction. Similarly, the 3Pi 6 Mbps dataset, being normally distributed with very high kurtosis, also contributes to easier predictability. LR models generally performed poorly, and FFNNs were typically the worst performers. This indicates that worst-case percentiles are more predictable, which is a favourable outcome for their application with DDS. RF and XGB are identified as the most suitable models for these scenarios. Furthermore, linear transformations are highly likely to outperform logarithmic transformations for worst-case percentiles, and often

for normal-case percentiles as well.

Dataset	Model	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi No	LR	-0.0098	-0.0063	0.0245	-0.0061	–
	RF	-10.3114	0.3408	0.3171	0.3315	–
	XGB	0.6779	0.7021	0.8685	0.7644	–
	FFNN	-0.0054	-0.0058	-0.0014	-0.0060	–
3Pi 6 Mbps	LR	0.0938	-0.0437	0.1416	-0.0335	linear
	RF	0.9320	0.7509	0.9380	0.8840	linear
	XGB	-0.2816	0.6419	0.9294	0.7883	–
	FFNN	-0.0444	-0.0153	-0.0450	-0.0184	log
3Pi 8 Mbps	LR	-0.0206	-0.1518	0.0725	-0.0431	linear
	RF	-0.1981	0.2963	0.8898	0.4339	–
	XGB	-2.9436	-0.3927	0.8617	0.2124	–
	FFNN	-0.3471	-0.4791	-0.0410	-0.0399	–
5Pi No	LR	-26.3795	0.0811	-0.1337	0.0294	–
	RF	-10.6874	0.1863	-0.3103	0.0903	–
	XGB	-0.8080	0.1091	-0.1498	0.0800	log
	FFNN	-10.9065	-0.1432	-0.0276	-0.0119	–
5Pi 24 Mbps	LR	0.2181	-0.0859	0.2240	-0.0776	linear
	RF	0.5592	0.4963	0.5862	0.4564	linear
	XGB	0.5885	0.3618	0.5982	0.3964	linear
	FFNN	0.0037	-0.0708	0.0094	-0.0639	linear
5Pi 71 Mbps	LR	-13.1312	-0.1266	0.0992	-0.0379	–
	RF	-50.1062	0.3624	0.4943	0.3668	–
	XGB	-155.9156	0.5082	0.4884	0.6091	–
	FFNN	-6.7350	-3.4458	0.0386	0.0057	–

Table 5.23: Latency R^2 Comparison: Linear vs. Logarithmic Transformations for All Percentiles Sorted by Dataset

All Percentiles Analysis Table 5.23 presents the R^2 values for linear and logarithmic transformations across all percentiles for various models and datasets. The results indicate that linear transformations outperform logarithmic transformations in 7 out of 24 cases, logarithmic transformations outperform linear transformations in only 2 cases, and in the remaining 15 cases, neither outperforms the other. This suggests that the choice of transformation is not universally applicable and may depend on the specific dataset and model used.

Table 5.24 summarises the cases where linear transformations outperform logarithmic transformations for all percentiles. All FFNN R^2 values are negative, rendering them unhelpful for performance prediction. For XGB, although logarithmic transformations generally outperform linear transformations, the results remain poor, with a mean R^2 of 0.11 and a median R^2 of 0.08. Beyond this specific instance, logarithmic transformations do not statistically significantly outperform linear transformations.

Dataset	Model	Linear Mean	Log Mean	Linear Median	Log Median
3Pi 6 Mbps	LR	0.0938	-0.0437	0.1416	-0.0335
	RF	0.9320	0.7509	0.9380	0.8840
3Pi 8 Mbps	LR	-0.0206	-0.1518	0.0725	-0.0431
5Pi 24 Mbps	LR	0.2181	-0.0859	0.2240	-0.0776
	RF	0.5592	0.4963	0.5862	0.4564
	XGB	0.5885	0.3618	0.5982	0.3964
	FFNN	0.0037	-0.0708	0.0094	-0.0639

Table 5.24: Cases where Linear Transformation Outperforms Logarithmic Transformation for All Percentiles for Latency

Linear transformations generally outperform logarithmic transformations as seen in Table 5.25. FFNNs consistently perform very poorly, even worse than LR models. For the 5Pi 24 Mbps dataset, XGB performed better with linear transformations, while RF performed better with logarithmic transformations, which was contrary to initial expectations. Therefore, RF and XGB remain the best-performing models. For the 5Pi 24 Mbps dataset, linear transformations generally yield better results than logarithmic transformations, despite the DDS performance distribution showing a heavy-tailed distribution for linear and a normal distribution for logarithmic. Notably, logarithmic transformations do not outperform linear transformations for any of the well-performing models.

Model	Dataset	Linear Mean	Log Mean	Linear Median	Log Median
FFNN	3Pi 6 Mbps	-0.0444	-0.0153	-0.0450	-0.0184
XGB	5Pi No	-0.8080	0.1091	-0.1498	0.0800

Table 5.25: Cases where Logarithmic Transformation Outperforms Linear Transformation for All Percentiles for Latency

Throughput

Just like with latency, we remove any results where both linear and log were negative to focus on cases where at least one transformation yielded a positive R^2 score. The analysis of throughput results is broken down into three main parts: analysis of normal-case percentiles and worst-case percentiles, followed by all percentiles.

Normal and Worst-Case Percentiles Analysis Tables 5.26, 5.27, 5.28, and 5.29 present the R^2 values for linear and logarithmic transformations across normal-case percentiles for LR, RF, XGB, and FFNN models respectively. RF and XGB models consistently outperformed all other models across various scenarios. This superior performance is attributed to their

nature as ensemble methods, which are highly effective at identifying and modelling intricate, non-linear relationships within the data.

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi 6 Mbps	Normal	0.6905	0.3636	0.7006	0.3990	linear
	Worst	0.5382	0.1433	0.5485	0.1495	linear
3Pi 8 Mbps	Normal	0.6798	0.5122	0.6316	0.4818	linear
	Worst	0.4240	0.1798	0.4226	0.1720	linear
5Pi No	Normal	0.4100	0.2050	0.4093	0.2036	linear
	Worst	0.2928	0.0592	0.3030	0.0660	linear

Table 5.26: Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for FFNN Models

Conversely, FFNNs and LR models consistently yielded the poorest results. While FFNNs possess the inherent capability to model non-linearities, their sub-optimal performance suggests either a lack of optimal tuning or that the underlying data patterns were not sufficiently complex for FFNNs to demonstrate a clear advantage over the tree-based models. LR’s consistently poor predictions indicate that the relationship within the throughput data is likely non-linear, fundamentally violating the model’s assumptions.

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi 6 Mbps	Normal	0.3453	0.2664	0.3410	0.2808	linear
	Worst	0.2257	0.0462	0.2298	0.0536	linear
3Pi 8 Mbps	Normal	0.4151	0.3164	0.3925	0.2977	linear
	Worst	0.2323	0.0157	0.2322	0.0186	linear
3Pi No	Normal	0.5840	-1.2861	0.5847	-1.2898	linear
	Worst	0.5774	-1.4783	0.5752	-1.3481	linear
5Pi 24 Mbps	Normal	0.4465	0.3652	0.4660	0.3790	linear
	Worst	0.2351	0.0764	0.2515	0.0802	linear
5Pi No	Normal	0.3761	0.1982	0.3702	0.1823	linear
	Worst	0.2733	0.0441	0.2909	0.0604	linear

Table 5.27: Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for LR Models

Models generally exhibit better performance for normal-case percentiles compared to worst-case percentiles. This is primarily because normal-case percentiles datasets typically contain larger value ranges and more data instances, providing richer information for model training. Notably, the 5Pi 71 Mbps dataset generally yielded very high R^2 scores for normal-case percentiles, with RF achieving a linear mean R^2 of 0.9636. Despite having the largest number of samples, the 3Pi No dataset did not consistently show the best performance, particularly for

FFNN and LR models, although RF performed exceptionally well. This suggests that the sheer size of a dataset is less critical than its inherent complexity and noise characteristics.

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
3Pi 6 Mbps	Normal	0.7152	0.6413	0.6857	0.6016	linear
	Worst	0.5737	0.5018	0.5870	0.5042	linear
3Pi No	Normal	0.8893	0.8817	0.8905	0.8837	linear
	Worst	0.8604	0.8295	0.8731	0.8408	linear
5Pi 71 Mbps	Normal	0.9636	0.9448	0.9680	0.9524	linear
	Worst	0.8164	0.7471	0.8690	0.7847	linear
5Pi No	Normal	0.7663	0.7415	0.7691	0.7378	linear
	Worst	0.5774	0.5230	0.6515	0.5738	–

Table 5.28: Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for RF Models

Anomalously, LR models exhibited negative R^2 values for the 3Pi No dataset. A negative R^2 indicates that the model performs worse than simply predicting the mean of the dependent variable. This phenomenon could be attributed to model misspecification, where LR is fundamentally unsuitable for the log-transformed data in these cases, potentially due to a highly non-linear relationship or the introduction of properties by the log transformation that violate the assumptions of LR.

Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
5Pi 71 Mbps	Normal	0.9339	0.8851	0.9361	0.9088	linear
	Worst	0.8144	0.7065	0.8913	0.7358	linear
5Pi No	Normal	0.7391	0.7176	0.7296	0.7099	linear
	Worst	0.5170	0.4235	0.5571	0.4725	linear

Table 5.29: Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for Normal-Case Percentiles for XGB Models

As seen in Table 5.30, in all observed cases, logarithmic transformations only outperformed linear transformations for normal-case percentiles never for worst-case percentiles. XGB generally achieved the best results, particularly with the 3Pi No Throttle dataset. FFNN performance varied, but these models constituted the majority of instances where logarithmic transformations surpassed linear ones.

Comparing normal-case percentiles and worst-case percentiles, normal-case percentiles consistently yielded superior results. This is expected, as normal-case percentiles represent the more prevalent and stable segment of the data distribution, while worst-case percentiles are sparser, making pattern recognition more challenging. XGB performed exceptionally well with the 3Pi No dataset, likely due to the abundance of samples available. FFNNs demonstrated

Model	Dataset	Percentile Scope	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
FFNN	3Pi No	Normal	0.6532	0.6787	0.6481	0.6825	log
		Worst	0.6402	0.5991	0.6380	0.5901	linear
	5Pi 24 Mbps	Normal	0.8385	0.8605	0.8429	0.8663	log
		Worst	0.6649	0.6595	0.6888	0.6773	–
	5Pi 71 Mbps	Normal	0.5133	0.7487	0.5138	0.7270	log
		Worst	0.3602	0.4510	0.4170	0.4588	–
XGB	3Pi No	Normal	0.9035	0.9240	0.9044	0.9257	log
		Worst	0.8677	0.8253	0.8914	0.8916	–

Table 5.30: Throughput R^2 Comparison: Logarithmic vs. Linear Transformations for Normal vs. Worst Case Percentiles

particularly strong performance with the 5Pi 24 Mbps normal-case percentiles dataset, further enhanced by a logarithmic transformation. Notably, none of these specific results were negative indicating that the models were able to capture the underlying patterns effectively.

Model	Linear Count	Log Count	Neither Count
LR	11	0	1
RF	8	0	4
XGB	5	1	6
FFNN	7	3	2
Total	31	4	13

Table 5.31: Number of cases where Linear vs. Logarithmic Transformations are Most Performant for Throughput R^2 by Model

The majority of cases (14 out of 24) indicate that linear transformations outperform logarithmic transformations as shown in Table 5.31. In 6 out of 24 cases, neither transformation significantly outperformed the other. Only 4 cases showed logarithmic transformations outperforming linear ones, with 3 of these instances involving FFNNs. Notably, none of these results yielded negative values. Normal-case percentiles consistently exhibit better results, as they represent a larger and more prevalent portion of the dataset, facilitating pattern recognition and modelling. Therefore, if using FFNNs with larger datasets (training sizes ranging from 342 to 841 samples), improved performance is likely to be observed with a logarithmic transformation.

All Percentiles Analysis Tables 5.33 and 5.34 present the R^2 values for linear and logarithmic transformations across all percentiles for all models. For FFNNs, linear transformation results are consistently poor, with a mean R^2 of 0.0037 and a median R^2 of 0.0094. This performance is even worse than that of LR models. This sub-optimal performance is likely due to the learning rate never exceeding 0.1, which may hinder the model’s ability to learn complex patterns effectively.

Dataset	Linear Count	Log Count	Neither Count
3Pi No	2	2	0
3Pi 6 Mbps	3	0	1
3Pi 8 Mbps	2	0	2
5Pi No	4	0	0
5Pi 24 Mbps	1	1	2
5Pi 71 Mbps	2	1	1

Table 5.32: Number of cases where Linear vs. Logarithmic Transformations are Most Performant for Throughput R^2 by Dataset

Model	Dataset	Linear Mean	Log Mean	Linear Median	Log Median	Most Performant
LR	3Pi No	0.5824	-1.3319	0.5844	-1.2939	linear
	3Pi 6 Mbps	0.3169	0.2139	0.3158	0.1693	linear
	3Pi 8 Mbps	0.3716	0.2448	0.3427	0.2028	linear
	5Pi No	0.3516	0.1615	0.3698	0.1720	linear
	5Pi 24 Mbps	0.3962	0.2964	0.4293	0.3086	linear
	5Pi 71 Mbps	0.3278	0.1010	0.3512	0.3009	–
RF	3Pi No	0.8824	0.8693	0.8886	0.8768	linear
	3Pi 6 Mbps	0.6815	0.6081	0.6561	0.5803	linear
	3Pi 8 Mbps	0.6412	0.6459	0.6403	0.6639	–
	5Pi No	0.7214	0.6895	0.7677	0.7328	linear
	5Pi 24 Mbps	0.8664	0.8252	0.9045	0.8739	linear
	5Pi 71 Mbps	0.9285	0.8977	0.9648	0.9429	–
XGB	3Pi No	0.8950	0.9005	0.9036	0.9249	–
	3Pi 6 Mbps	0.6214	0.6076	0.5814	0.5934	–
	3Pi 8 Mbps	0.6164	0.6021	0.5862	0.5070	–
	5Pi No	0.6862	0.6476	0.7248	0.7061	–
	5Pi 24 Mbps	0.8684	0.8339	0.8871	0.8546	linear
	5Pi 71 Mbps	0.9054	0.8426	0.9334	0.8824	linear
FFNN	3Pi No	0.6501	0.6597	0.6465	0.6369	–
	3Pi 6 Mbps	0.6542	0.3111	0.6490	0.3019	linear
	3Pi 8 Mbps	0.6189	0.4330	0.5770	0.3954	linear
	5Pi No	0.3821	0.1703	0.3992	0.1880	linear
	5Pi 24 Mbps	0.7972	0.8127	0.8343	0.8450	–
	5Pi 71 Mbps	0.4769	0.6778	0.5014	0.7113	log

Table 5.33: Throughput R^2 Comparison: Linear vs. Logarithmic Transformations for All Percentiles

Considering the 5Pi 24 Mbps dataset, it was anticipated that XGB would exhibit greater sensitivity to data transformations and perform better with logarithmic transformations. However, the results show the opposite: RF’s logarithmic R^2 values (mean of 0.4963, median of 0.4564) are superior to XGB’s (mean of 0.3618, median of 0.3964). Conversely, it was expected that RF, being less sensitive to transformations, would outperform XGB in linear results. Yet, XGB’s linear R^2 mean of 0.5885 surpasses RF’s mean of 0.5592, and XGB’s linear median of 0.5982 also outperforms RF’s median of 0.5862. These observations contradict typical expecta-

Model	Dataset	Linear Mean	Log Mean	Linear Median	Log Median
LR	3Pi No	0.5824	-1.3319	0.5844	-1.2939
	3Pi 6 Mbps	0.3169	0.2139	0.3158	0.1693
	3Pi 8 Mbps	0.3716	0.2448	0.3427	0.2028
	5Pi No	0.3516	0.1615	0.3698	0.1720
	5Pi 24 Mbps	0.3962	0.2964	0.4293	0.3086
RF	3Pi No	0.8824	0.8693	0.8886	0.8768
	3Pi 6 Mbps	0.6815	0.6081	0.6561	0.5803
	5Pi No	0.7214	0.6895	0.7677	0.7328
	5Pi 24 Mbps	0.8664	0.8252	0.9045	0.8739
XGB	5Pi 24 Mbps	0.8684	0.8339	0.8871	0.8546
	5Pi 71 Mbps	0.9054	0.8426	0.9334	0.8824
FFNN	3Pi 6 Mbps	0.6542	0.3111	0.6490	0.3019
	3Pi 8 Mbps	0.6189	0.4330	0.5770	0.3954
	5Pi No	0.3821	0.1703	0.3992	0.1880

Table 5.34: Cases where Linear Transformation Outperforms Logarithmic Transformation for All Percentiles (Throughput)

tions regarding the sensitivity of these models to data transformations. RF and XGB models generally achieved the highest R^2 scores. This superior performance is attributed to their ensemble-based nature, which excels at capturing intricate, non-linear relationships within the data. Conversely, FFNNs exhibited very mediocre performance. While FFNNs possess the capability to learn non-linear patterns, their R^2 scores were consistently lower than those of both tree-based ensemble methods. This suggests potential for more extensive fine-tuning or that the underlying relationships in the data were not sufficiently complex for FFNNs to outperform RF or XGB. LR models were consistently poor performers, indicating that the relationships within the data are likely non-linear. Anomalously, LR models displayed negative R^2 values for the 3Pi No dataset.

There was only one instance where modelling in log space outperformed modelling in linear space for all percentiles, observed with the FFNN model on the 5Pi 71 Mbps dataset. This specific case had a linear mean of 0.4769 and a logarithmic mean of 0.6778, with corresponding medians of 0.5014 and 0.7113 respectively.

In summary, RF and XGB models consistently demonstrated superior performance. FFNNs exhibited very mediocre results while LR models were consistently poor performers and there was only one instance where a logarithmic transformation outperformed a linear transformation.

5.6.5 Single Versus Combined Output

The prediction of latency and throughput can be modelled either independently or concurrently. Each metric’s distribution is characterised by 23 percentiles. Consequently, models can be configured to predict either 23 percentiles for a single metric or all 46 percentiles (23 for each metric) simultaneously. For clarity, predicting 23 percentiles for a single metric is termed “single”, whilst predicting all 46 percentiles is referred to as “combined”. It was hypothesised that single-output models would exhibit superior performance compared to combined-output models, given the increased complexity associated with simultaneously predicting multiple, potentially disparate, outputs.

To determine statistical significance when comparing between the two (single versus combined), a multi-step approach was employed. The W Test was initially conducted to assess the normality of the data distributions. In instances where neither distribution was found to be normal, Welch’s t-test was subsequently applied as the significance test. Conversely, if one or both distributions exhibited normality, the U Test and the KS Test were used. Following the determination of statistical significance, a comparative analysis of means and medians was performed to ascertain the direction of any observed differences.

Statistical significance was conclusively determined based on the following criteria:

- Both distributions exhibiting normality, coupled with significant results from both Welch’s t-test and the KS Test.
- Both distributions being non-normal, with significant results from both the U Test and the KS Test.
- One or both distributions exhibiting normality, alongside significant results from both the U Test and the KS Test.

This systematic approach ensured a statistically robust conclusion regarding whether single-output or combined-output modelling yielded superior performance.

Latency

We remove cases where both single and combined results were negative from the analysis, as these cases do not provide meaningful insights into the performance of the models. For the majority of cases (25 out of 43), neither transformation demonstrated superior performance. In 14 cases, combined-output models outperformed single-output models, while in 4 cases, single-output models were superior.

As anticipated, tree-based models consistently outperformed all other models, owing to

their proficiency in handling non-linear patterns. Conversely, FFNNs and LR exhibited poor performance, often yielding negative results, particularly for combined or linear outputs. Logarithmic transformations (log1p) generally improved results for XGB and RF with skewed datasets, while linear transformations proved more effective in specific instances, such as the 5Pi 24 Mbps dataset for RF. Single output models frequently outperformed combined output models for XGB and RF. However, combined output models achieved superior results in certain cases, suggesting the presence of dataset-specific patterns. FFNNs consistently struggled, indicating a poor model fit, likely attributable to insufficient tuning or small dataset sizes.

Model	Dataset	Output Transformation	Single Mean	Combined Mean	Single Median	Combined Median
LR	3Pi No	none	-0.0087	0.2681	0.0241	0.2385
	3Pi 6 Mbps	log1p	-0.0431	0.0864	-0.0335	-0.0152
		none	0.1001	0.2056	0.1484	0.1837
	3Pi 8 Mbps	none	-0.0100	0.1784	0.0788	0.1737
	5Pi No	log1p	0.0739	0.1149	0.0054	0.1548
	5Pi 24 Mbps	log1p	-0.0902	0.1039	-0.0905	-0.0057
		none	0.2264	0.3109	0.2590	0.3269
RF	3Pi No	log1p	0.3463	0.7893	0.3454	0.8571
	3Pi 8 Mbps	log1p	0.3209	0.5130	0.4466	0.5495
	5Pi 24 Mbps	log1p	0.5178	0.6570	0.5874	0.7752
		none	0.5714	0.6962	0.6152	0.7332
XGB	5Pi 24 Mbps	log1p	0.3857	0.6293	0.4153	0.7189
FFNN	5Pi No	log1p	-0.1325	0.1241	-0.0124	0.1333
	5Pi 24 Mbps	log1p	-0.0751	0.2066	-0.0728	0.0493

Table 5.35: Cases where combined output outperforms single output for latency.

Cases where combined outperforms single Table 5.35 presents the cases where combined output models outperformed single output models for latency. Among the models where combined output outperformed single output, LR accounted for 7 instances, RF for 4, FFNN for 2, and XGB for 1. This suggests that LR is particularly effective at modelling combined metric behaviour, implying a linear relationship between latency and throughput. Specifically, for LR, 7 out of 10 cases demonstrated superior combined output performance, with the remaining 3 showing no significant difference. Therefore, when employing LR, modelling combined outputs is recommended for improved results.

The 5Pi 24 Mbps dataset was observed in 6 instances where combined output outperformed single output. Its frequent appearance suggests a potentially stronger and more easily identifiable relationship between latency and throughput compared to other datasets. Conversely,

other datasets, including 5Pi No, 3Pi 6 Mbps, 3Pi 8 Mbps, and 3Pi No, each appeared only twice. Notably, the 5Pi 71 Mbps dataset was never observed in these cases, possibly indicating a less identifiable relationship between its metrics or insufficient data for the model to capture it effectively. For context, the 5Pi 71 Mbps dataset comprises 342 training samples and 73 validation samples, while the 3Pi 6 Mbps dataset has 186 training samples and 40 validation samples. This raises the question of whether the relationship for 3Pi 6 Mbps is inherently simpler to identify than for 5Pi 71 Mbps.

Regarding output transformations, the log1p transformation was observed 9 times, while no transformation was used in 5 instances. This higher frequency of log1p suggests its beneficial relationship with combined modelling. Log1p transformation effectively reduces the disparity between throughput and latency values, making them more amenable to combined modelling. In contrast, a linear transformation would exaggerate this gap, especially for worst-case values where throughput is very small and latency is very large, making it challenging to model both metrics together in linear space. Thus, if modelling with log1p, combined output is often preferred, and vice versa.

Combined mean R^2 values ranged from 0.0864 to 0.7893. RF models achieved the best combined mean R^2 of 0.7893 with the 3Pi No dataset and log1p transformation, and the worst of 0.5130 with the 3Pi 8 Mbps dataset and log1p transformation. LR models yielded the worst combined mean R^2 results, with a best of 0.3109 for the 5Pi 24 Mbps dataset with a linear transformation, and a worst of 0.0864 for the 3Pi 6 Mbps dataset with a log transformation.

Combined median R^2 values ranged from -0.0152 to 0.8571. RF models again achieved the best combined median R^2 of 0.8571 with the 3Pi No dataset and log transformation, and the worst of 0.5495 with the 3Pi 8 Mbps dataset and log transformation. LR models had the worst combined median R^2 results, with a best of 0.3269 for the 5Pi 24 Mbps dataset with a linear transformation, and a worst of -0.0152 for the 3Pi 6 Mbps dataset with a log transformation.

It is generally unexpected for combined output models to outperform single output models due to the increased complexity. However, when this occurs, it is likely attributable to a strong correlation between latency and throughput. Additionally, a larger dataset probably aids the model in learning complex patterns for 46 outputs.

Cases where single outperforms combined Table 5.36 presents the cases where single output models outperformed combined output models for latency. Only 4 cases demonstrated statistically significant superior performance for single output models over combined output models. These instances exclusively involved tree-based models: 2 cases for RF and 2 for

XGB. This suggests that for certain datasets, the complexity introduced by combining outputs might be better handled by the models when predicting metrics individually. The datasets involved were 3Pi 6 Mbps (3 cases) and 5Pi 24 Mbps (one case). Both log1p and linear output transformations were equally represented, with 2 cases each.

Model	Dataset	Output Transformation	Single Mean	Combined Mean	Single Median	Combined Median
RF	3Pi 6 Mbps	log1p	0.7688	0.6204	0.9063	0.5870
		none	0.9359	0.5760	0.9454	0.6162
XGB	3Pi 6 Mbps	log1p	0.6679	0.5043	0.8105	0.5689
	5Pi 24 Mbps	none	0.6005	-65.1283	0.6310	-18.8169

Table 5.36: Cases where Single Output Outperforms Combined Output for Latency

Anomalously, XGB with the 5Pi 24 Mbps dataset exhibited incredibly poor performance for combined output, with a mean R^2 of -65.1283 and a median R^2 of -18.8169. For single output models, mean R^2 values ranged from 0.6005 to 0.9359, and median R^2 values ranged from 0.6310 to 0.9454. In contrast, for combined output models, mean R^2 values ranged from -65.1283 to 0.6204, and median R^2 values ranged from -18.8169 to 0.6162. Such significantly negative results for combined output, particularly from robust models like XGB and RF, are unexpected. This phenomenon is likely attributable to the presence of extreme outliers or substantial noise within the data, which severely impacts the combined prediction.

Throughput

We removed one case where all results were negative (for both single and combined). Combined output models never outperform single output models and single output models outperform combined output models in 22 out of 47 cases while neither model outperforms the other in 25 out of 47 cases. The results are almost equally split between cases where neither model outperforms the other and cases where single output models outperform combined output models. Table 5.37 presents the summary of results for throughput and the cases where single output models outperform combined output models.

Cases where single outperforms combined Among the models, LR appeared 8 times, FFNN 7 times, XGB 4 times, and RF 3 times. LR was frequently observed, with 8 out of 11 cases showing single output outperforming combined output. FFNNs also exhibited this trend.

Regarding datasets, 5Pi 24 Mbps was the most frequent, appearing 7 times, followed by 3Pi 6 Mbps, 3Pi 8 Mbps, and 3Pi No, each appearing 4 times. 5Pi No appeared twice, and

Model	Dataset	Output Trans-formation	Single Mean	Combined Mean	Single Median	Combined Median
LR	3Pi No	none	0.5762	0.2681	0.5841	0.2385
	3Pi 6 Mbps	log1p	0.2159	0.0864	0.2012	-0.0152
	3Pi 6 Mbps	none	0.3110	0.2056	0.3158	0.1837
	3Pi 8 Mbps	log1p	0.2511	0.0540	0.2496	-0.0199
	3Pi 8 Mbps	none	0.3669	0.1784	0.3427	0.1737
	5Pi 24 Mbps	log1p	0.2980	0.1039	0.3086	-0.0057
	5Pi 24 Mbps	none	0.3954	0.3109	0.4293	0.3269
	5Pi 71 Mbps	log1p	0.0954	-0.0132	0.3009	-0.0279
RF	3Pi 8 Mbps	log1p	0.6391	0.5130	0.6639	0.5495
	5Pi 24 Mbps	log1p	0.8243	0.6570	0.8739	0.7752
	5Pi 24 Mbps	none	0.8631	0.6962	0.9045	0.7332
XGB	3Pi No	log1p	0.8990	0.6870	0.9184	0.8185
	5Pi No	log1p	0.6252	0.2156	0.7061	0.1392
	5Pi 24 Mbps	log1p	0.8291	0.6293	0.8546	0.7189
	5Pi 24 Mbps	none	0.8580	-65.1283	0.8871	-18.8169
FFNN	3Pi No	log1p	0.6521	0.0002	0.6289	-0.0086
	3Pi No	none	0.6414	0.2491	0.6460	-0.0013
	3Pi 6 Mbps	log1p	0.3112	0.0013	0.3019	-0.0326
	3Pi 6 Mbps	none	0.6492	-1.5509	0.6490	-0.0739
	3Pi 8 Mbps	log1p	0.4417	0.1044	0.4376	0.0218
	5Pi No	log1p	0.1644	0.1241	0.1844	0.1333
	5Pi 24 Mbps	log1p	0.8060	0.2066	0.8450	0.0493

Table 5.37: Cases where single output outperforms combined output for throughput.

5Pi 71 Mbps only once. The frequent appearance of 5Pi 24 Mbps, which is the second largest dataset, suggests that the relationship between latency and throughput for this dataset is likely the least correlated.

For output transformations, log1p was observed 14 times, linear 8 times, FFNN 5 times, LR 4 times, XGB 3 times, and RF 2 times. Logarithmic transformations were more prevalent than linear transformations, which is expected as they generally improve predictions for skewed data.

Mean performance for single output models ranged from 0.0954 to 0.8990, while combined output models ranged from -65.1283 to 0.6962. Median performance for single output models ranged from 0.1844 to 0.9184, and for combined output models, it ranged from -18.8169 to 0.8185.

RF and XGB models consistently demonstrated superior performance, achieving high R^2 values for single metric predictions (e.g., RF for latency on 3Pi 6 Mbps with no transformation yielded 0.9359; XGB for throughput on 3Pi No with log1p transformation yielded 0.8990). These models effectively capture non-linear patterns within the data.

FFNNs exhibited poor performance for latency prediction, often resulting in negative R^2 values (e.g., -9.9478 for 5Pi No with no transformation and single output). However, FFNNs showed better performance for throughput (e.g., 0.8060 for 5Pi 24 Mbps with log1p transformation and single output), suggesting that throughput is an easier metric to model.

LR models consistently performed as the weakest, yielding negative or very low R^2 values for latency (e.g., -24.0753 for 5Pi No with no transformation and single output) and only modest R^2 values for throughput (e.g., 0.5762 for 3Pi No with no transformation and single output).

Single metric prediction (23 outputs) generally outperformed combined metric prediction (46 outputs), particularly for tree-based models and FFNNs (e.g., XGB for throughput on 5Pi 24 Mbps with no transformation yielded 0.8580 for single output versus -65.1283 for combined output).

Regarding transformations, no transformation was often more effective for throughput prediction (e.g., RF for 5Pi 24 Mbps with no transformation yielded 0.8631 versus 0.8243 with log1p transformation). Conversely, log1p transformation sometimes improved latency prediction (e.g., RF for 3Pi 6 Mbps with log1p transformation yielded 0.7688).

Expected observations included the excellence of tree-based models, with RF and XGB achieving R^2 values greater than 0.8 for single metrics in larger datasets (e.g., 3Pi No, 5Pi 24 Mbps), confirming their ability to capture non-linearities. Throughput was generally easier to predict, evidenced by higher R^2 values (e.g., FFNN for 5Pi 24 Mbps with log1p transformation yielded 0.8060) compared to latency (e.g., -0.0751), likely due to less skewness or simpler underlying patterns. Single output models were expected to outperform combined output models due to their lower complexity, resulting in better R^2 values (e.g., RF for latency on 3Pi 6 Mbps with no transformation yielded 0.9359 for single output versus 0.5760 for combined output). FFNNs were anticipated to struggle with latency, often yielding negative R^2 values (e.g., -3.3824 for 5Pi 71 Mbps with log1p transformation and single output), possibly due to insufficient tuning or complex latency distributions. LR's poor performance, characterised by low or negative R^2 values (e.g., latency on 5Pi 71 Mbps with no transformation yielded -11.9684), further confirmed its unsuitability for non-linear, high-dimensional outputs.

Unexpected cases included instances where combined output models outperformed single output models. For example, RF for latency on 3Pi 8 Mbps with log1p transformation showed combined output (0.5130) outperforming single output (0.3209), suggesting that latency/throughput correlation might aid combined prediction. Similarly, RF for latency on 3Pi No with log1p transformation yielded combined output (0.7893) significantly higher than single output (0.3463), possibly due to the large dataset (841 training samples) enabling the learning

of 46 outputs. RF for latency on 5Pi 24 Mbps with both log1p and no transformations also showed combined output (0.6570 with log1p, 0.6962 with no transformation) outperforming single output (0.5178 with log1p, 0.5714 with no transformation), indicating potential synergistic patterns. XGB for latency on 5Pi 24 Mbps with log1p transformation also exhibited combined output (0.6293) outperforming single output (0.3857), an unexpected result where correlated outputs might be beneficial. Even FFNN for latency on 5Pi 24 Mbps with log1p transformation showed combined output (0.2066) outperforming single output (-0.0751), a rare instance of positive R^2 for FFNN, suggesting log1p stabilises combined outputs.

Negative R^2 values were unexpectedly observed in tree-based models. XGB for latency on 5Pi 71 Mbps with no transformation yielded extremely poor results for both single (-142.2992) and combined (-66.8640) outputs, suggesting extreme outliers or scaling issues. XGB for throughput on 3Pi No with no transformation and combined output also showed a severely negative R^2 of -309.1640, indicating a severe data issue for untransformed combined outputs. RF for latency on 5Pi 71 Mbps with no transformation also produced unexpected negative R^2 values for single (-45.6848) and combined (-14.2692) outputs, which is unusual for a robust model and likely points to noise or outliers.

FFNNs showed unexpected competitiveness for throughput prediction. For 5Pi 24 Mbps with log1p transformation and single output, FFNN achieved an R^2 of 0.8060, close to tree-based models, suggesting throughput data is more predictable for FFNNs. Similarly, for 3Pi No with log1p transformation and single output, FFNN achieved a strong R^2 of 0.6521, indicating that large datasets (841 training samples) aid FFNN learning.

LR unexpectedly showed combined output outperforming single output for latency across multiple datasets (e.g., 3Pi 6 Mbps, 5Pi 24 Mbps), with combined output (e.g., 0.3109 for 5Pi 24 Mbps with no transformation) outperforming single output (e.g., 0.2264). This unexpected behaviour might be due to overfitting or correlated outputs.

No transformation unexpectedly outperformed log1p transformation in some cases. For throughput with RF on 5Pi 24 Mbps, no transformation ($R^2 = 0.8631$) outperformed log1p ($R^2 = 0.8243$), suggesting throughput data is less skewed than anticipated. For latency with RF on 3Pi 6 Mbps, no transformation ($R^2 = 0.9359$) outperformed log1p ($R^2 = 0.7688$), which is unusual if latency is skewed and might indicate data symmetry.

In conclusion, RF and XGB are the best models for both metrics, with throughput generally being easier to predict. Single metric prediction is more reliable, while combined prediction struggles with negative R^2 in some cases. No transformation is often better for throughput, whereas log1p helps latency in skewed datasets. FFNNs show promise for throughput but re-

quire extensive tuning for latency. Negative R^2 values in tree-based models (e.g., 5Pi 71 Mbps, 3Pi No) indicate preprocessing issues. It is recommended to use XGB for both metrics, prioritising single metric prediction with `log1p` for latency and no transformation for throughput. Further investigation into 5Pi 71 Mbps and 3Pi No datasets is needed to address outliers or scaling issues causing negative R^2 . FFNNs should be tuned for throughput in larger datasets (e.g., 3Pi No) but avoided for latency unless heavily optimised. Finally, analysing latency/throughput correlations is crucial to explain cases where combined output unexpectedly outperforms single output, and modelling latency and throughput separately could improve combined prediction reliability.

Summary

RF and XGB models consistently demonstrated superior performance, achieving high R^2 values for single metric predictions. FFNNs exhibited poor performance for latency prediction but showed better performance for throughput, suggesting that throughput is an easier metric to model. LR models consistently performed as the weakest. Single metric prediction generally outperformed combined metric prediction, particularly for tree-based models and FFNNs. Regarding transformations, no transformation was often more effective for throughput prediction, while `log1p` transformation sometimes improved latency prediction. Expected observations included the excellence of tree-based models and throughput being generally easier to predict. Unexpected cases included instances where combined output models outperformed single output models for latency, and negative R^2 values in tree-based models. FFNNs showed unexpected competitiveness for throughput prediction. LR unexpectedly showed combined output outperforming single output for latency. No transformation unexpectedly outperformed `log1p` transformation in some cases. In conclusion, RF and XGB are the best models for both metrics, with throughput generally being easier to predict. Single metric prediction is more reliable, while combined prediction struggles with negative R^2 in some cases. No transformation is often better for throughput, whereas `log1p` helps latency in skewed datasets. FFNNs show promise for throughput but require extensive tuning for latency. It is recommended to use XGB for both metrics, prioritising single metric prediction with `log1p` for latency and no transformation for throughput. Further investigation into datasets causing negative R^2 is needed. FFNNs should be tuned for throughput in larger datasets but avoided for latency unless heavily optimised. Finally, analysing latency/throughput correlations is crucial to explain cases where combined output unexpectedly outperforms single output.

5.7 Discussion

5.7.1 Initial discussion

The initial phase of this research focused on establishing the feasibility and effectiveness of employing ML models for DDS performance prediction. This involved a comparative analysis of LR and RF models, evaluated across both interpolation and extrapolation scenarios. Performance was quantified using the RMSE and R^2 .

For LR, the models generally exhibited sub-optimal performance. In interpolation, the highest R^2 achieved was 0.54 for average throughput, with several models falling below an R^2 of 0.2. Extrapolation performance for LR was notably inferior, with only a few models surpassing an R^2 of 0.2. However, an interesting insight emerged with metrics such as “Total Throughput”, where LR demonstrated a 26% improvement in R^2 (from 0.37 to 0.47) and a 23% reduction in RMSE during extrapolation compared to interpolation. This suggests a potential for LR in specific extrapolation contexts, particularly for metrics with a high frequency of zero values, such as “Lost Samples”, where the model’s prediction of zero values proved advantageous. Despite these isolated instances of better extrapolation, the overall performance of LR models indicated their limited suitability for accurately modelling DDS performance metrics.

In stark contrast, RF models consistently demonstrated superior performance across all metrics and scenarios. In interpolation, nine out of eleven RF models achieved an R^2 greater than 0.9, with the “average received samples” model reaching a near-perfect R^2 of 0.999. Even the lowest-performing RF model for interpolation (latency) still achieved an R^2 of 0.636, significantly outperforming LR counterparts. For extrapolation, five RF models maintained an R^2 between 0.9 and 1, highlighting their robust generalisation capabilities. RF models consistently outperformed LR models in both interpolation and extrapolation, underscoring their greater suitability for modelling DDS performance.

An important insight from this initial work is the critical role of the RRMSE in interpreting model accuracy, as RMSE in isolation can be misleading. While RF models generally excelled, some metrics, such as “Latency” and “Total Lost Samples”, presented challenges, exhibiting lower R^2 values and higher RRMSEs, particularly in extrapolation. This suggests that metrics with high variance or a predominance of zero values are inherently more difficult to predict accurately.

In conclusion, the initial results unequivocally established the feasibility and utility of employing ML for DDS performance prediction, with RF models proving to be a robust and

effective choice. This foundational work paved the way for further, more detailed investigations into the impact of network noise, data transformations, and multi-output predictions.

5.7.2 Interpolation Versus Extrapolation

The investigation into interpolation versus extrapolation aimed to ascertain the conditions under which ML models could effectively predict performance beyond their training data range. This analysis specifically sought to identify instances where extrapolation yielded an R^2 value of 0.8 or greater, signifying robust predictive capability.

Key findings revealed that extrapolation performed well in 17 specific cases. Notably, the majority of these successful extrapolations were observed for throughput metrics (8 cases), followed by latency (5 cases), and combined metrics (4 cases). RF models were the most prevalent in these high-performing extrapolation scenarios, accounting for 9 instances, whilst XGB models contributed 6 instances, and FFNNs accounted for the remaining 2. This prevalence of tree-based models in successful extrapolation was unexpected, as these models are generally not anticipated to excel in predicting values outside their training distribution.

Further analysis indicated that successful extrapolation was frequently associated with the application of a `log1p` output transformation (14 out of 17 cases) and the 5Pi 24Mbps dataset (9 out of 17 cases). The `log1p` transformation likely contributed by normalising skewed data distributions, thereby making the data more amenable to modelling and extrapolation. The frequent appearance of the 5Pi 24Mbps dataset suggests that its inherent data patterns are particularly conducive to generalisation by the models.

An intriguing aspect of this work was the identification of cases where extrapolation unexpectedly outperformed interpolation. Four such instances were observed, exclusively involving tree-based models (RF and XGB). These differences in R^2 values, ranging from 0.0064 to 0.0336, were statistically insignificant, necessitating cautious interpretation. However, the occurrence of such cases, particularly with tree-based models, challenges conventional assumptions regarding their extrapolation capabilities. Further investigation into the input distributions revealed an overlap in “Data Length” and “Publisher Count” features between the validation and out-of-sample datasets for the 5Pi 24Mbps dataset, which likely contributed to these favourable extrapolation results. This suggests that when out-of-distribution data shares similar input characteristics with the training data, even tree-based models can generalise effectively.

In summary, whilst interpolation generally yields superior performance, this section demonstrated that effective extrapolation is achievable under specific conditions, particularly with

tree-based models, appropriate data transformations, and datasets exhibiting amenable underlying patterns. The unexpected success of tree-based models in extrapolation, albeit in limited and statistically insignificant cases, warrants further investigation to fully comprehend the underlying mechanisms facilitating this behaviour.

Potential future work stemming from these findings includes a more granular analysis of how individual input parameters influence extrapolation performance. Specifically, investigating the impact of separating interpolation and extrapolation based on single input parameters (e.g., splitting by data length or publisher count) could provide deeper insights into the generalisation capabilities of these models.

5.7.3 Network Noise

The impact of network noise on model performance was rigorously investigated through the introduction of varying bandwidth caps. This analysis aimed to understand how different levels of network constraint affect the predictive accuracy and robustness of the ML models.

Initial data exploration revealed a substantial degradation in model performance with the introduction of any bandwidth cap. Specifically, the average R^2 values for “Low” and “High” caps were significantly lower than for the “No” cap scenario, challenging the intuitive assumption that less noise would invariably lead to better R^2 scores. Furthermore, bandwidth caps led to considerably less consistent model performance, as evidenced by much higher standard deviations in R^2 values under capped conditions. Statistical tests, including ANOVA and Kruskal-Wallis, confirmed that bandwidth cap levels measurably and significantly impact model performance in the vast majority of cases.

An intriguing and counter-intuitive finding emerged from the analysis of performance extremes: the “Low” bandwidth cap surprisingly often yielded the highest mean and median R^2 scores. This suggests that a moderate level of noise may act as a beneficial regulariser, potentially preventing models from overfitting to the specific conditions of uncapped training data and thereby improving generalisation. Conversely, the “No” cap, whilst capable of the best performance, also frequently led to the lowest mean and median R^2 scores, indicating a risk of poor performance in unconstrained environments.

Analysis by testbed highlighted the influence of system scale on model predictability. On the more complex 5Pi testbed, the “No Cap” scenario was overwhelmingly the worst-performing, suggesting that unconstrained network traffic from a larger number of nodes leads to saturation and unpredictable behaviour, making performance harder to model. In this environment, the

“Low” cap most frequently achieved the highest mean R^2 , reinforcing the notion that moderate noise can regularise models in complex systems.

The intrinsic nature of the performance metric also played a crucial role. For latency prediction, moderate noise levels (e.g., “Low” cap) were most beneficial, likely due to their ability to regularise the long tail of high-value outliers often present in latency data. In contrast, throughput prediction appeared to benefit from more constrained environments, with the “High” cap often producing the best results. This indicates that a stable data rate, enforced by a cap, removes erratic fluctuations and creates a more consistent pattern for models to learn.

Finally, the response to network noise was not uniform across different ML models. Ensemble tree-based models, such as RF and XGB, demonstrated greater resilience and even benefited from moderate noise, suggesting their inherent robustness against certain noise characteristics. FFNNs and LR models, however, showed greater sensitivity to uncapped environments and were more susceptible to volatility.

In conclusion, the impact of network noise is a complex interplay between system scale, metric characteristics, and model architecture. Optimal noise mitigation strategies are deeply contextual, requiring a nuanced understanding of these interactions to achieve robust DDS performance prediction.

Future work should further explore the role of moderate noise as a regulariser in ML models.

5.7.4 Linear Versus Log Space

The analysis of linear versus logarithmic transformations for modelling performance metrics revealed a nuanced impact on both interpretability and predictive accuracy. It was hypothesised that linear transformations would yield higher R^2 values for worst-case latency percentiles due to their inherent variance, whilst logarithmic transformations would prove superior for normal-case percentiles by normalising skewed distributions and reducing outlier influence.

For latency, the results indicated that RF and XGB models consistently achieved high R^2 scores, particularly for worst-case percentiles, with RF notably reaching 0.9962 for the 3Pi 6Mbps dataset. This strong performance is attributed to their ensemble averaging and iterative error correction mechanisms, which are well-suited for complex, non-linear latency patterns. LR models generally performed poorly, especially for 5Pi datasets, suggesting their inability to capture the inherent non-linearities. FFNNs typically exhibited the poorest performance, potentially due to overfitting, insufficient hyperparameter tuning, or a constrained learning rate.

An interesting finding for latency was observed with the 3Pi No dataset, where logarithmic transformations surprisingly outperformed linear transformations for both RF and XGB models, particularly for worst-case percentiles. This suggests that for heavy-tailed distributions, log transformation can be more effective in capturing underlying patterns. However, for the majority of cases across all percentiles, neither transformation consistently outperformed the other, or linear transformations were superior, indicating that the optimal choice is highly dependent on the specific dataset and model.

For throughput, RF and XGB models again consistently outperformed other models, reinforcing their efficacy in modelling intricate, non-linear relationships. FFNNs and LR models yielded the poorest results, with LR often exhibiting negative R^2 values, indicating its unsuitability for the non-linear throughput data. Normal-case percentiles consistently showed better predictability than worst-case percentiles, likely due to larger value ranges.

Similar to latency, linear transformations generally outperformed logarithmic transformations for throughput, particularly for normal-case percentiles. However, FFNNs were the primary exception, with logarithmic transformations occasionally yielding better results, especially for larger datasets. Notably, there was only one instance where a logarithmic transformation outperformed a linear transformation for all throughput percentiles (FFNNs on 5Pi 71Mbps dataset).

In conclusion, the choice of transformation is not universally applicable and depends on the specific metric, dataset, and model. RF and XGB models are generally robust to both linear and logarithmic transformations, often performing well regardless. However, for highly skewed data like latency, logarithmic transformations can be beneficial, especially for normal-case percentiles. For throughput, linear transformations often prove more effective.

Future work could involve a more detailed investigation into the characteristics of datasets where anomalous results were observed (e.g., negative R^2 values for robust models or unexpected performance of transformations). Further exploration into dynamic transformation selection based on data characteristics could also be beneficial.

5.7.5 Single Versus Combined Output

The analysis of single versus combined output modelling explored whether predicting latency and throughput independently or concurrently yielded superior performance. It was hypothesised that single-output models would outperform combined-output models due to the increased complexity associated with simultaneously predicting multiple, potentially disparate, outputs.

For latency, the results indicated that in the majority of cases (58%), neither single nor combined output models demonstrated superior performance. As anticipated, tree-based models (RF and XGB) consistently outperformed other models, showcasing their proficiency in handling non-linear patterns. Conversely, FFNNs and LR models exhibited poor performance, often yielding negative results. While single-output models frequently outperformed combined-output models for XGB and RF, combined-output models achieved superior results in a significant number of cases (33%), suggesting the presence of dataset-specific patterns. Notably, LR models were particularly effective at modelling combined metric behaviour, implying a linear relationship between latency and throughput. The 5Pi 24Mbps dataset frequently appeared in instances where combined output outperformed single output, suggesting a potentially stronger and more easily identifiable relationship between latency and throughput in this dataset. The $\log_{1p}$ transformation also proved beneficial for combined modelling by reducing the disparity between throughput and latency values.

Conversely, for throughput, combined-output models never outperformed single-output models. The results were almost equally split between cases where neither model outperformed the other and cases where single-output models outperformed combined-output models (47%). LR, FFNN, and XGB models frequently showed single output outperforming combined output. The frequent appearance of the 5Pi 24Mbps dataset in these cases suggests that the relationship between latency and throughput for this dataset is likely the least correlated.

In summary, RF and XGB models consistently demonstrated superior performance for both metrics. While single-metric prediction generally proved more reliable, combined prediction struggled with negative R^2 values in some cases. The choice of transformation also played a role, with no transformation often being more effective for throughput, whilst $\log_{1p}$ helped latency in skewed datasets.

Unexpectedly, combined-output models occasionally outperformed single-output models for latency, particularly with tree-based models and LR, suggesting that strong latency/throughput correlations might aid combined prediction, especially in larger datasets. Conversely, negative R^2 values were unexpectedly observed in tree-based models for combined outputs, indicating potential issues with extreme outliers or scaling. FFNNs showed unexpected competitiveness for throughput prediction, suggesting that throughput data is more predictable for these models.

Future work should prioritise XGB for both metrics, favouring single-metric prediction with $\log_{1p}$ for latency and no transformation for throughput. Further investigation into datasets exhibiting negative R^2 values is needed to address outliers or scaling issues. Analysing latency/throughput correlations in greater detail is crucial to explain cases where combined output

unexpectedly outperforms single output, and modelling latency and throughput separately could improve combined prediction reliability.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has presented a comprehensive investigation into the performance of the DDS, employing a dual-pronged approach of empirical evaluation and model-based prediction. The primary objective was to systematically analyse the intricate relationships between DDS configuration parameters, network conditions, and the resultant performance in terms of latency and throughput. The research has successfully demonstrated that the performance of DDS is a complex interplay of these factors, often yielding counter-intuitive results that defy simplistic theoretical assumptions. Moreover, this thesis has presented findings on the feasibility and efficacy of employing ML methodologies to accurately predict DDS for unseen configurations.

This thesis set out to quantitatively assess the impacts of key QoS parameters, data payload sizes, and network environmental factors on DDS latency and throughput. This objective was successfully met through extensive empirical evaluation, which provided a detailed characterisation of these performance determinants. A further aim was to determine the extent to which ML models could accurately predict DDS performance for untested configurations, and to identify the most effective models for this task. This was achieved by developing and evaluating a suite of predictive models, demonstrating the high efficacy of tree-based ensemble methods in particular. The research also sought to understand how network topology and noise affect DDS performance, and whether these effects could be accurately modelled and predicted. Through experimentation on diverse physical and virtualised testbeds, this work has elucidated these complex interactions and validated the capability of the developed models to account for such environmental factors. Ultimately, this thesis has delivered upon its primary goal of providing actionable insights and a robust predictive framework to facilitate the optimal design and

deployment of DDS-based real-time distributed systems.

A robust empirical foundation was established through a meticulously designed experimental methodology. The goal was to generate comprehensive and reliable datasets for understanding and predicting DDS performance. This was accomplished by constructing three distinct test environments, including two physical RPi clusters and a virtualised platform, to ensure the validity and reproducibility of the findings. A bespoke software suite, AutoPerf, was developed to automate the execution of over 20 systematic experimental campaigns, which explored a vast configuration space. The resulting raw data was then subjected to a rigorous, multi-stage processing pipeline to produce 17 curated, analysis-ready datasets.

The experimental performance evaluation of DDS provided a deep, empirical understanding of its behaviour. In virtualised environments, DDS demonstrated resilience to DDoS attacks, although latency was impacted. The presence of malicious subscribers was shown to linearly increase latency and eventually cap throughput, highlighting scalability limits. A key finding of this thesis was the profound and often counter-intuitive impact of network topology on MC performance. The research demonstrated that, contrary to theoretical expectations, MC communication does not universally outperform UC. In virtualised environments, UC was frequently more performant, likely due to the limitations of virtual switches. On a smaller physical testbed, an MEP was identified, where the performance benefits of MC diminished and were eventually surpassed by UC at higher participant counts. This exhaustion point was not observed on a larger, more robust testbed, which underscores the critical role of network infrastructure and its ability to handle MC traffic efficiently. A granular, quantile-based analysis further revealed that while MC is the dominant choice for throughput, its effect on latency is more complex; UC can provide better latency in high-latency, worst-case scenarios, particularly when using best-effort reliability. Unexpectedly, MC was also found to be competitive, and sometimes superior, in many-to-one communication scenarios, challenging traditional assumptions about its use cases. The investigation into the impact of data length on performance also yielded nuanced results that defy simple assumptions. For latency, while smaller data payloads (e.g., 100 B) were generally advantageous for best-case performance, larger payloads (e.g., 512 B and 1 KB) were often found to be more performant in high-latency, worst-case scenarios. This was significantly influenced by the number of publishers, with fewer publishers favouring larger data lengths for worst-case latency. For throughput, while the general trend of larger payloads leading to higher throughput held, a mid-range payload of 512 B frequently outperformed the larger 1 KB payload, suggesting the existence of an optimal payload size for maximising throughput under certain conditions. The number of publishers was again a key factor, with a

direct relationship observed: more publishers favoured larger data lengths for achieving higher throughput. These findings highlight that the selection of an optimal data length is not a straightforward decision and requires careful consideration of the specific operational context, including the number of participants and the desired performance characteristics. The introduction of network noise confirmed that performance degrades under constrained bandwidth, but also revealed complex trade-offs, such as a reduction in complete communication stalls at the cost of lower data rates. Finally, the investigation into faulty throughput measurements determined that zero-throughput readings were often an artefact of network flooding under best-effort reliability, rather than a complete failure of the DDS system.

The core contribution of this thesis was the development and rigorous evaluation of model-based performance prediction techniques, which established the feasibility and high efficacy of using ML to predict DDS performance. The results unequivocally demonstrated that tree-based ensemble models, particularly RF and XGB, can predict performance with high accuracy, significantly outperforming simpler models like LR and even more complex FFNNs, which struggled without extensive tuning. A key finding was that while interpolation is generally more accurate, effective extrapolation is achievable under specific conditions, particularly for throughput prediction using tree-based models with a logarithmic data transformation. The investigation into network noise yielded the counter-intuitive insight that moderate noise can act as a beneficial regulariser, improving model performance by constraining outliers in latency data and stabilising volatile throughput patterns. The choice of data transformation was also shown to be critical; modelling in linear space proved more effective for predicting the extreme values found in worst-case percentiles, whereas logarithmic space was better for the bulk of the data in normal-case percentiles. Finally, the comparison of single versus combined metric prediction revealed that while predicting metrics individually is generally more reliable, a combined approach can be surprisingly advantageous for latency prediction, likely due to the model leveraging strong underlying correlations between latency and throughput, although this approach also carries a higher risk of significant prediction errors.

In summary, this thesis makes several significant contributions to the field of DDS systems. It introduces a novel methodological framework that successfully integrates large-scale empirical evaluation with model-based performance prediction, providing a scalable and efficient alternative to traditional analysis methods. A comprehensive and unique empirical dataset has been generated, capturing DDS performance across thousands of configurations, which serves as a valuable resource for the research community. This work represents the first systematic application of ML to DDS performance prediction, demonstrating the high efficacy of tree-based

ensemble models. Finally, this research has produced novel insights into the complex and often counter-intuitive performance characteristics of DDS, particularly concerning the impact of network topology on MC communication and the nuanced effects of data length. These contributions provide both a robust foundation for future research and practical, data-driven guidance for the design and optimisation of DDS systems.

6.2 Future Work

Building upon the findings of this research, several promising avenues for future work have been identified. The insights gained from this thesis can serve as a springboard for further exploration into the performance of DDS and the refinement of predictive modelling techniques.

A planned but uncompleted avenue of research was the application of this thesis’s methodology to a real-world use case. A collaboration with the robotics department, as suggested by Peter Popov, was initiated to collect performance data from a drone operating on ROS2, which uses DDS as its underlying communication middleware. The objective was to apply our predictive modelling techniques to this real-life data, thereby validating the practical applicability and robustness of our framework in a dynamic, mission-critical environment. This work was unfortunately not completed due to time constraints but remains a compelling and important next step for future research.

The extensive data collection campaigns undertaken for the ML portion of this work have resulted in datasets that are significantly larger and more diverse than those used for the initial experimental performance evaluation. A valuable avenue for future work would be to revisit the analyses on the impact of MC and data length, applying the same methodologies to these newer, richer datasets. This would not only serve to validate the initial findings across a much wider range of thousands of new configurations but also potentially uncover more subtle performance characteristics and interactions that were not apparent in the smaller, original datasets.

Further advancements in predictive modelling could be achieved by exploring more sophisticated ML architectures. For instance, Diffusion Models could be employed to generate synthetic performance data, which would be invaluable for augmenting the training datasets, particularly for rare or hard-to-replicate high-load configurations. This would enhance the robustness and generalisation capabilities of the predictive models. Furthermore, the explicit network topology of the DDS system could be modelled using Graph Neural Networks (GNNs). This approach would allow the model to directly learn from the structure of the distributed system, potentially capturing complex dependencies between participants that are currently only implicitly repre-

sented in the data. Additionally, the application of Bayesian inference models could provide a probabilistic understanding of performance, quantifying the uncertainty associated with predictions, which is crucial for risk assessment in mission-critical systems. Ideally, future models would move beyond predicting discrete percentiles and instead learn a complete, continuous representation of the performance distributions. This would enable the full sampling of performance characteristics, offering a much richer and more nuanced prediction than is currently possible.

Another promising research direction is the inverse problem of performance prediction: real-time performance adaptation. This would involve developing models that, given a desired performance profile (e.g., a target latency and throughput), could recommend a set of optimal QoS parameter values. Such a system would effectively classify performance characteristics to determine the underlying configurations, enabling dynamic, on-the-fly optimisation of DDS systems in response to changing operational requirements.

Appendix A

Appendix A: Algorithmic Procedures

This appendix provides figures describing the algorithmic procedures used in the eCDF stochastic ordering method discussed in Subsection 3.3.4.

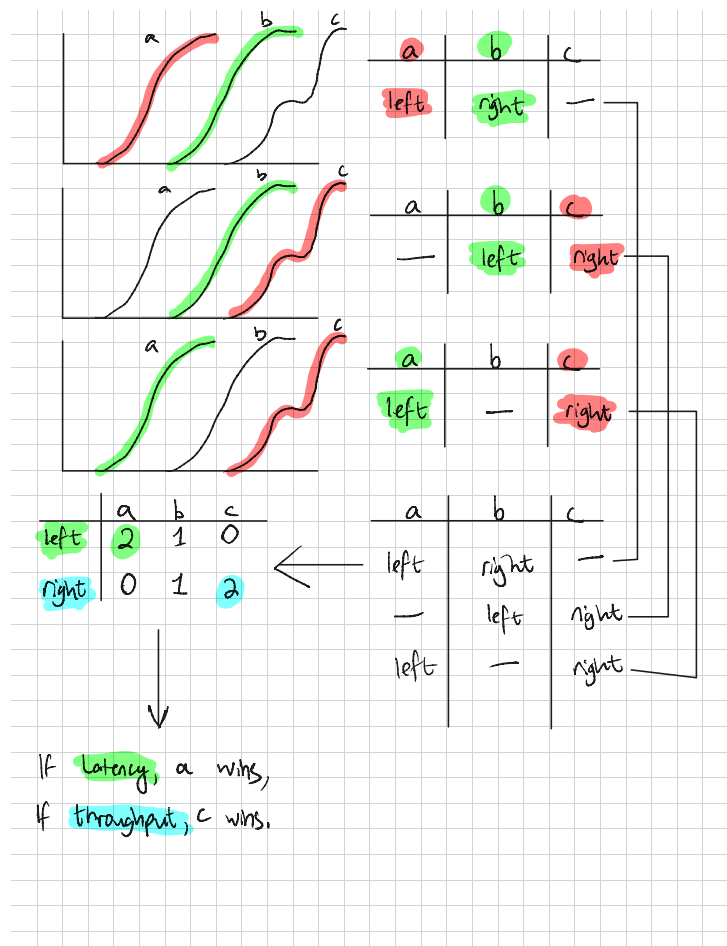


Figure A.1: Sketch of the algorithmic procedure for eCDF stochastic ordering. The left most and right most values are calculated per pairing. The value that appear left-/right-most majority of the time without intersecting with others is classified as the most performant.

Appendix B

Appendix B: Experimental Results

This appendix provides figures that supplement the experimental results presented in Chapter 4.

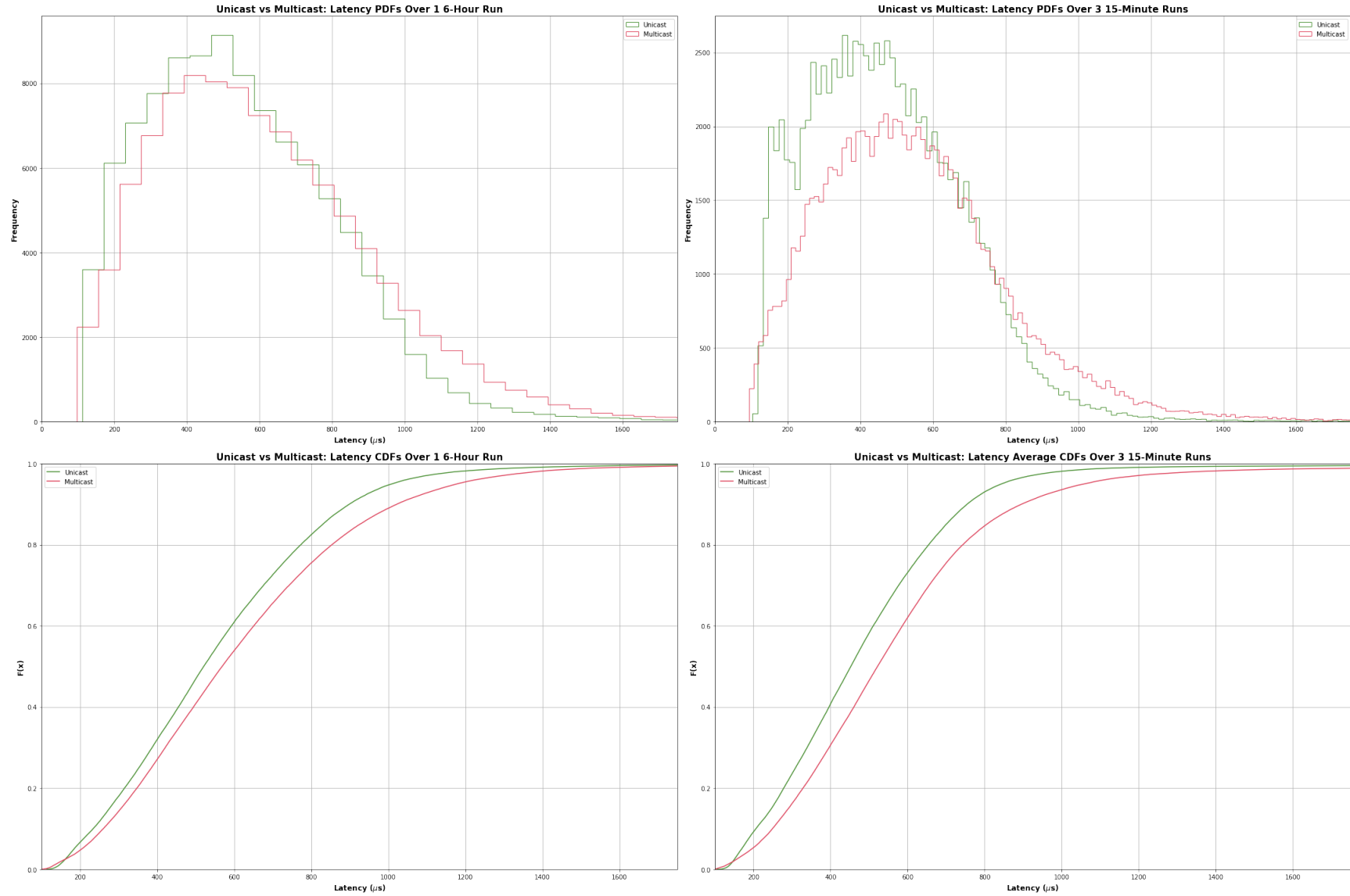


Figure B.1: Latency comparison between the 15-minute and 6-hour configurations. Top row: latency PDF. Bottom row: latency eCDF. Left column: 6 hour run. Right column: Latencies averages from 3 15-minute runs.

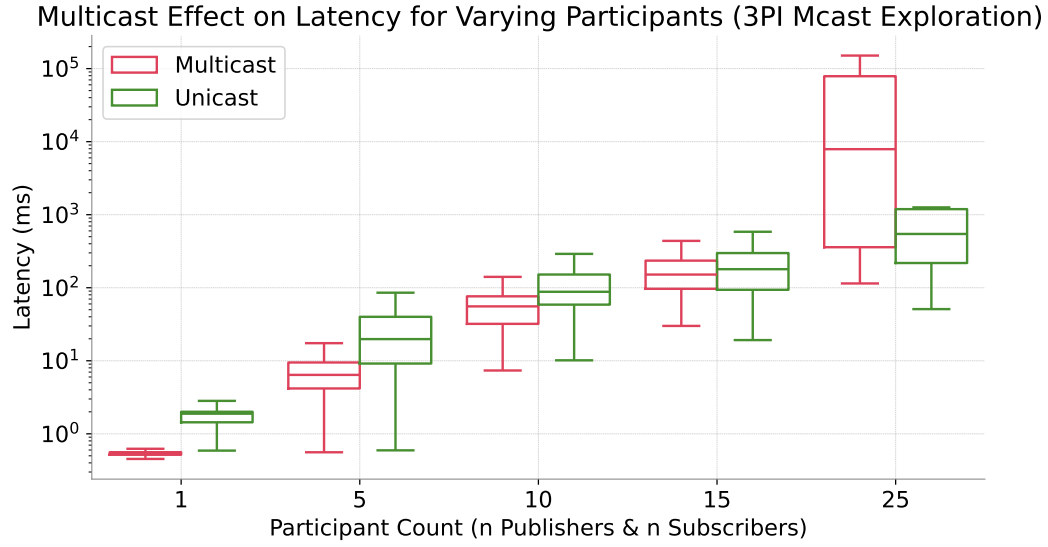


Figure B.2: Box plots comparing MC (red) to UC (green) latency for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.

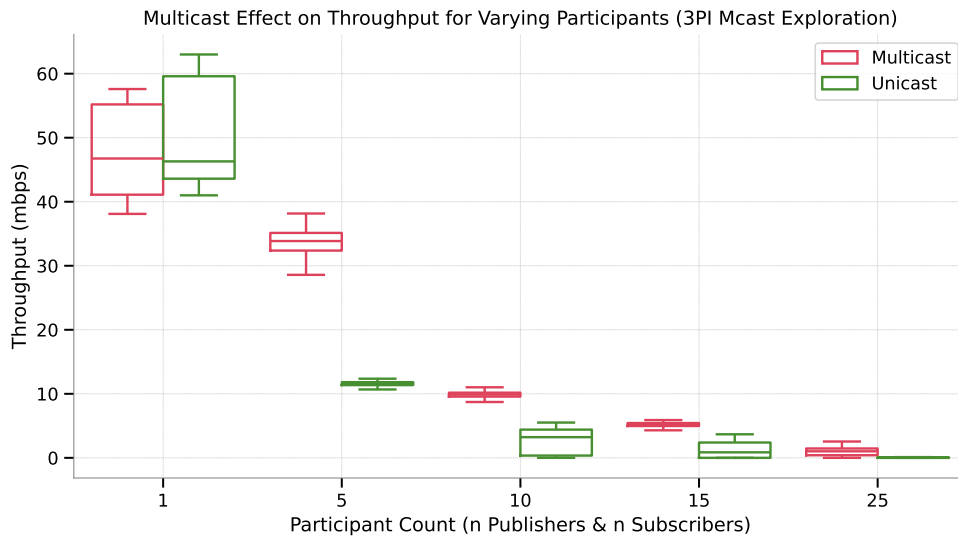


Figure B.3: Box plots comparing MC (red) to UC (green) throughput for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.

B.1 Participant Allocation Analysis on Multicast for 5Pi QoS Dataset

We define 3 participant allocations: - 1 for publishers and 3 for subscribers: “1P3S” - 2 for publishers and 2 for subscribers: “2P2S” - 3 for publishers and 1 for subscribers: “3P1S”

B.1.1 n Publishers and n Subscribers for all Participant Allocations

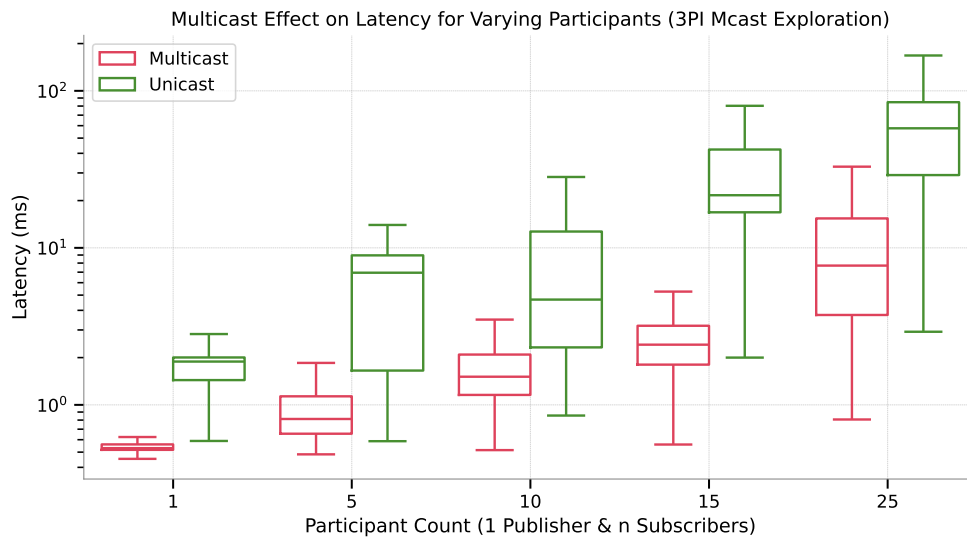


Figure B.4: Box plots comparing MC (red) to UC (green) latency for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.

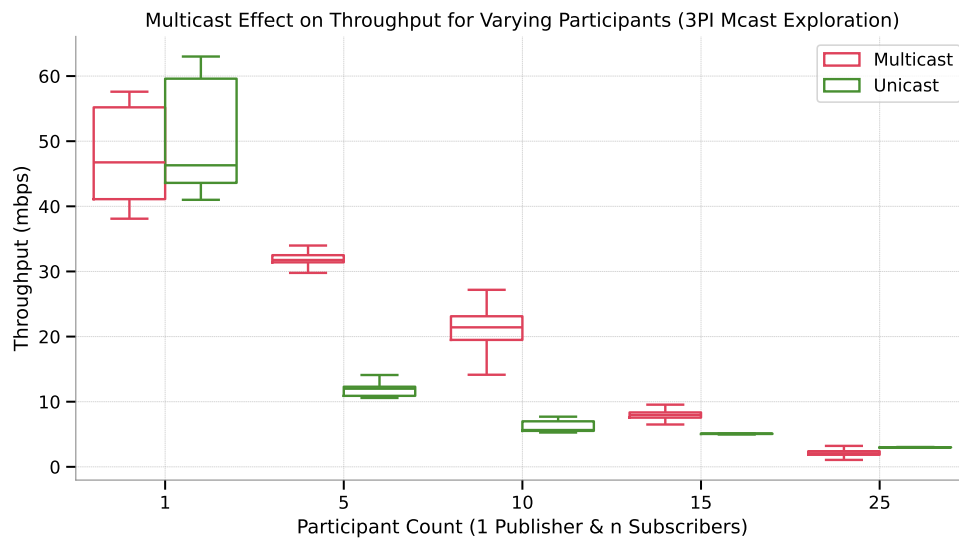


Figure B.5: Box plots comparing MC (red) to UC (green) throughput for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 3Pi Multicast Exploration.

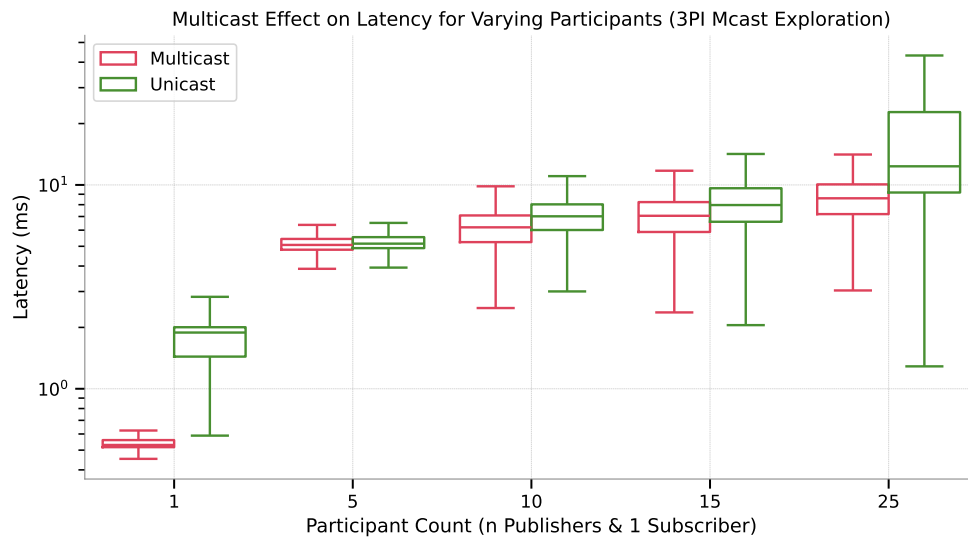


Figure B.6: Box plots comparing MC (red) to UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 3Pi Multicast Exploration.

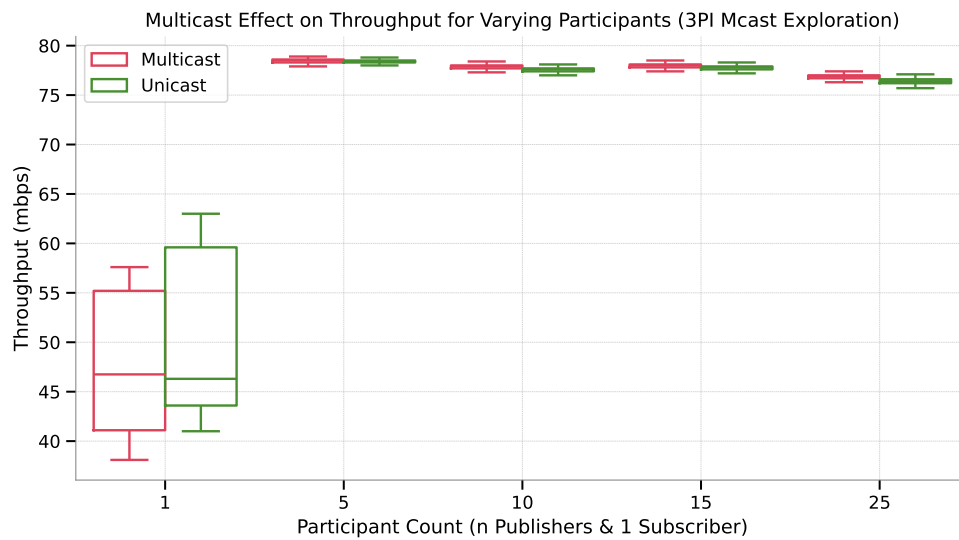


Figure B.7: Box plots comparing MC (red) to UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 3Pi Multicast Exploration.

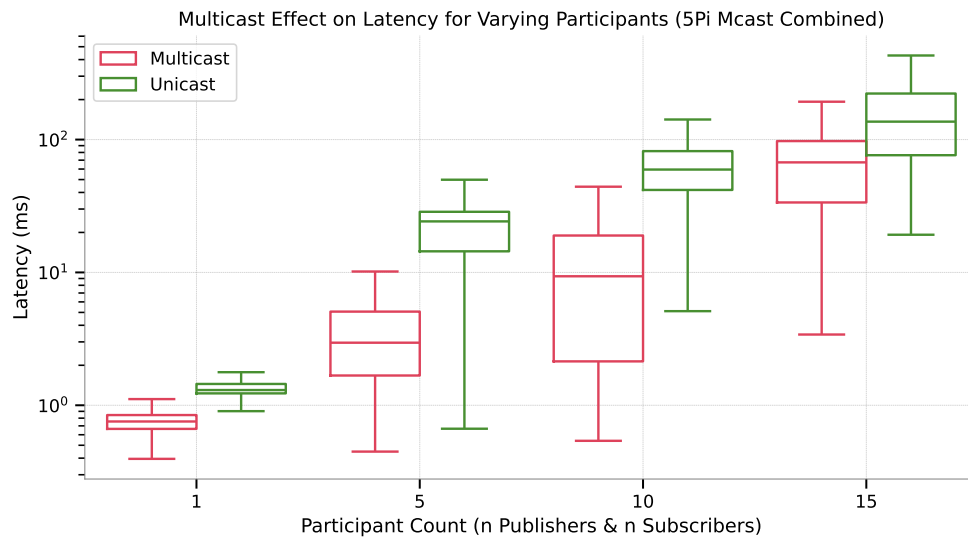


Figure B.8: Box plots comparing MC (red) to UC (green) latency for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.

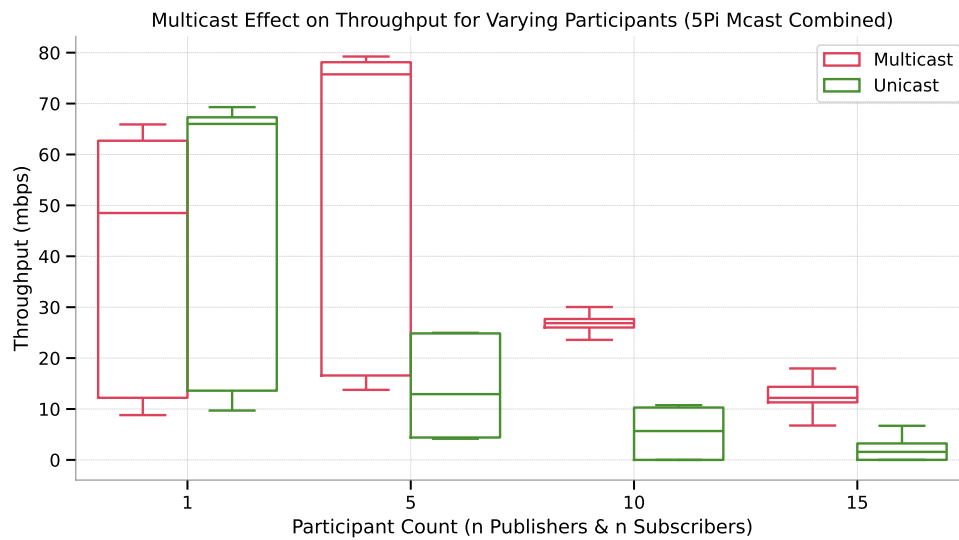


Figure B.9: Box plots comparing MC (red) to UC (green) throughput for 1, 5, 10, 15, and 25 publishers and subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.

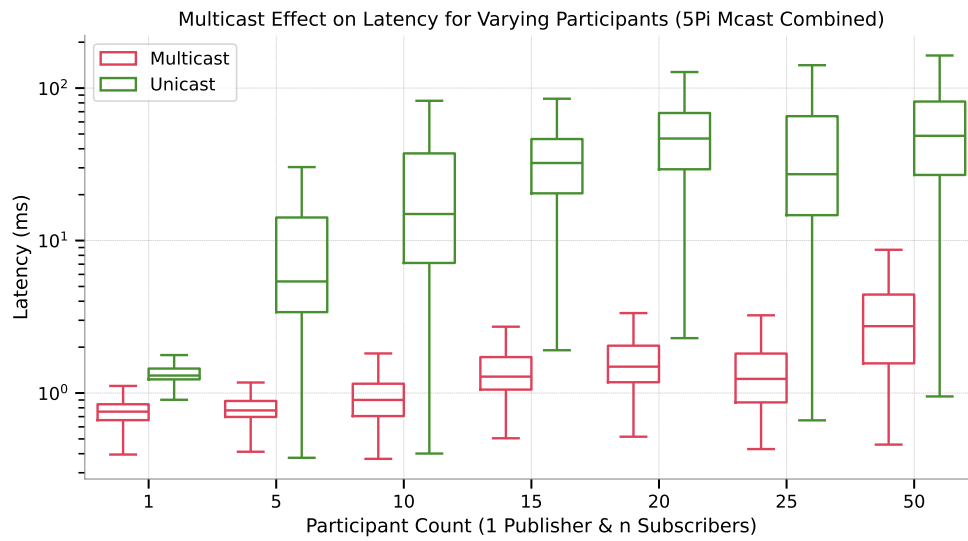


Figure B.10: Box plots comparing MC (red) to UC (green) latency for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.

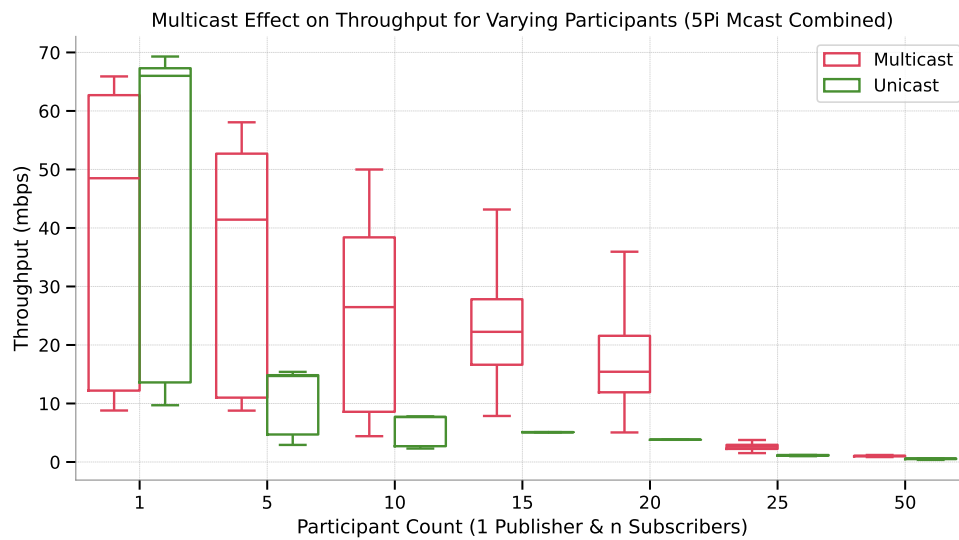


Figure B.11: Box plots comparing MC (red) to UC (green) throughput for 1 publisher and 1, 5, 10, 15, and 25 subscribers. The y-axis is in log scale. From 5Pi Multicast Exploration.

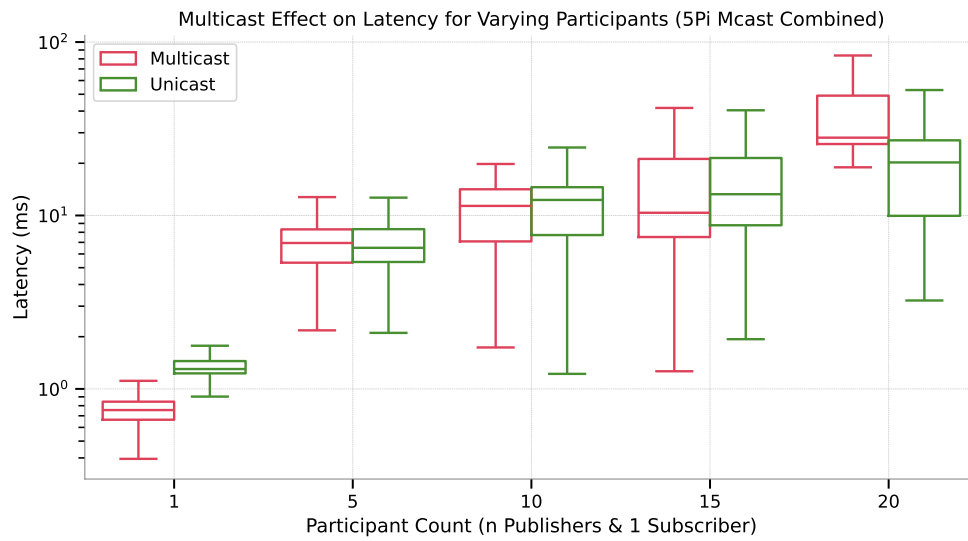


Figure B.12: Box plots comparing MC (red) to UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi Multicast Exploration.

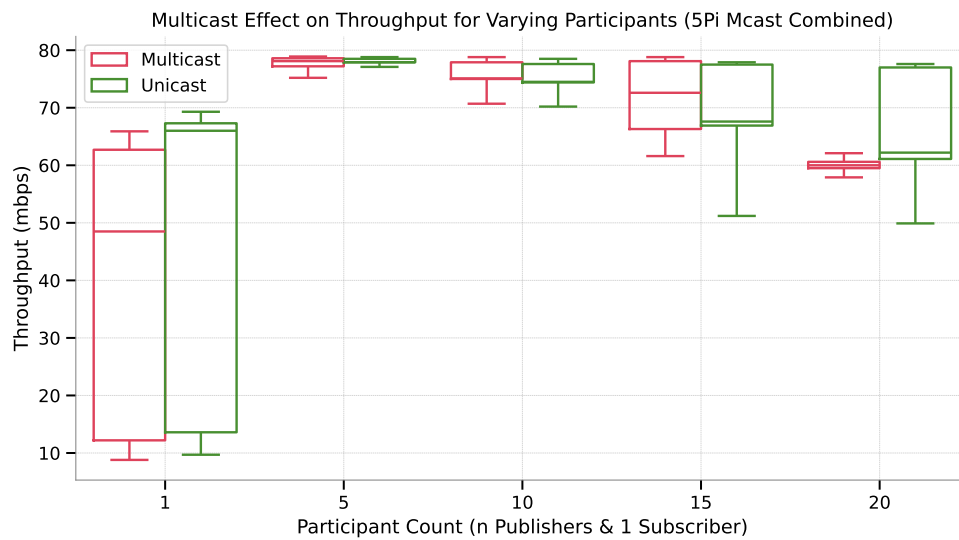


Figure B.13: Box plots comparing MC (red) to UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi Multicast Exploration.

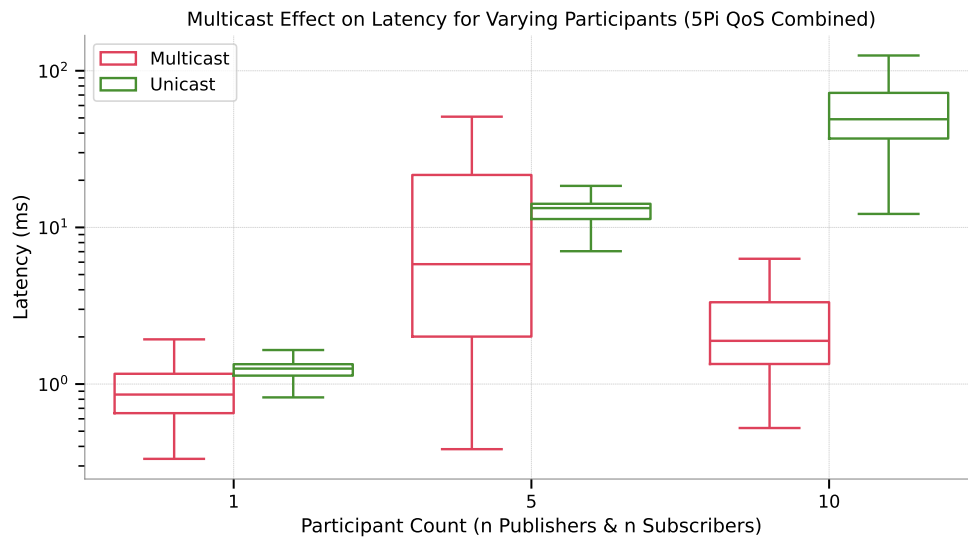


Figure B.14: Box plots comparing MC (red) to UC (green) latency for 1, 5, 10, and 15 publishers and subscribers. The y-axis is in log scale. From 5Pi QoS.

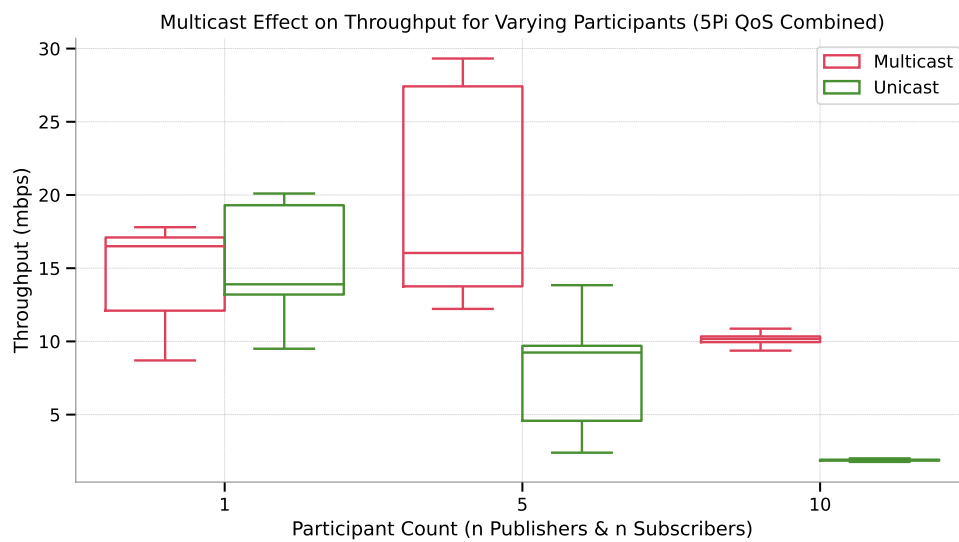


Figure B.15: Box plots comparing MC (red) to UC (green) throughput for 1, 5, 10, and 15 publishers and subscribers. The y-axis is in log scale. From 5Pi QoS.

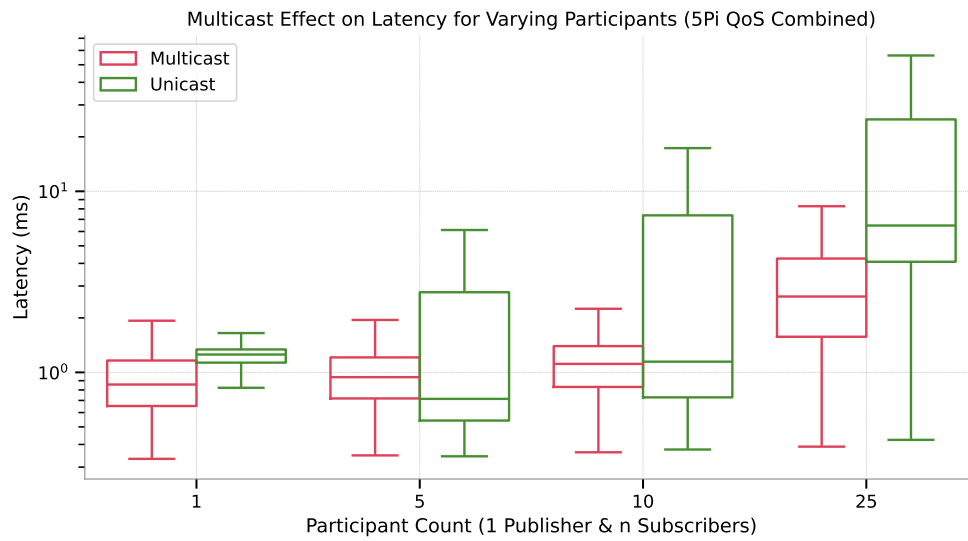


Figure B.16: Box plots comparing MC (red) to UC (green) latency for 1 publisher and 1, 5, 10, and 25 subscribers. The y-axis is in log scale. From 5Pi QoS.

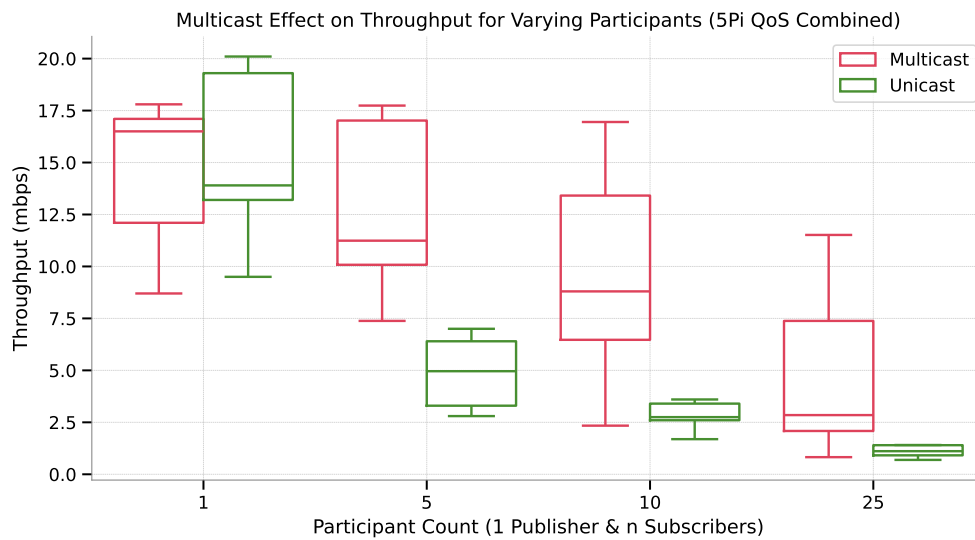


Figure B.17: Box plots comparing MC (red) to UC (green) throughput for 1 publisher and 1, 5, 10, and 25 subscribers. The y-axis is in log scale. From 5Pi QoS.

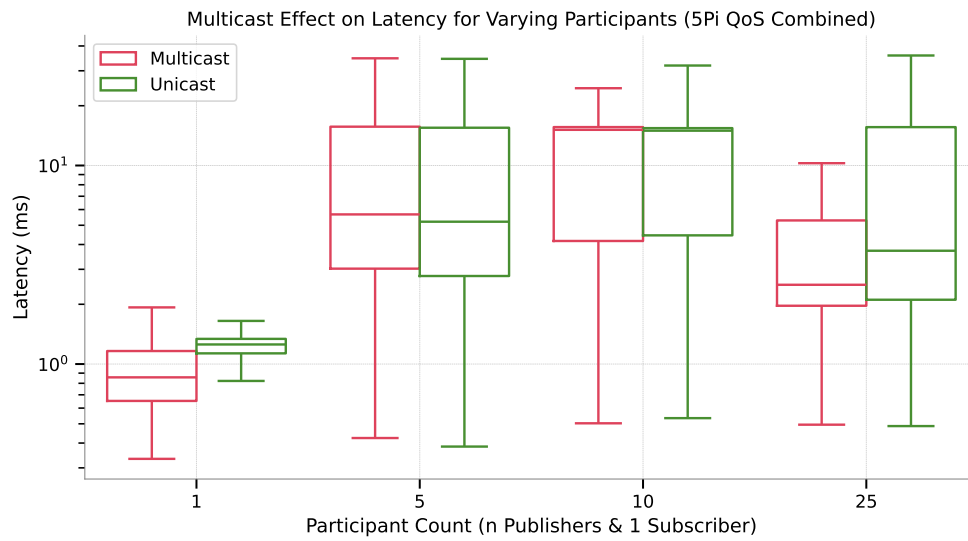


Figure B.18: Box plots comparing MC (red) to UC (green) latency for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi QoS.

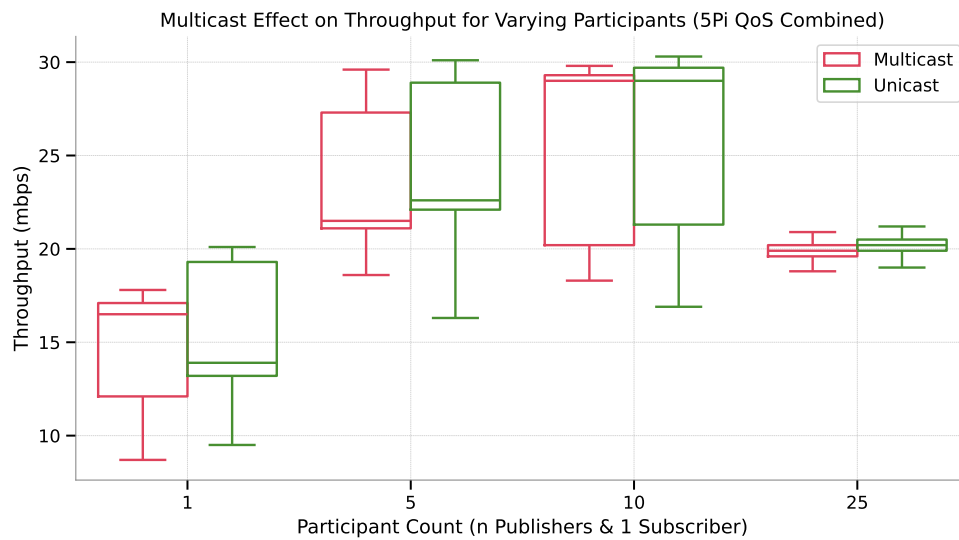


Figure B.19: Box plots comparing MC (red) to UC (green) throughput for 1 subscriber and 1, 5, 10, 15, and 25 publishers. The y-axis is in log scale. From 5Pi QoS.

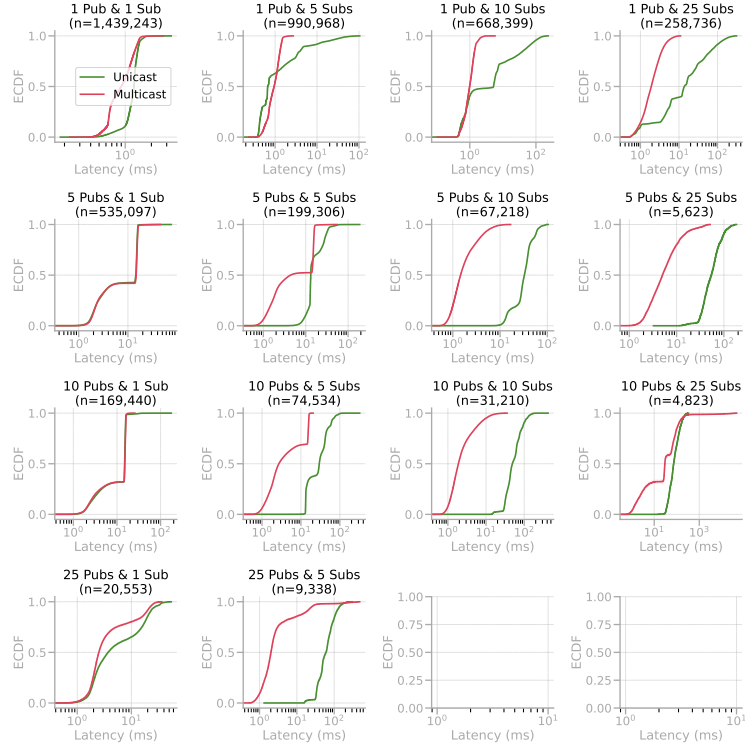


Figure B.20: eCDFs comparing UC (green) to MC (red) latency for the "1P3S" allocation. From 5Pi QoS.

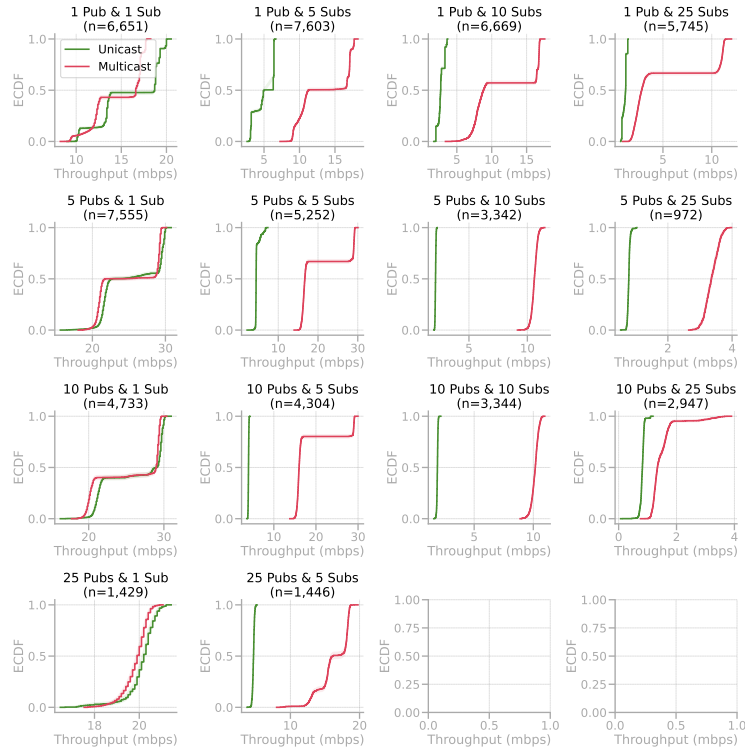


Figure B.21: eCDFs comparing UC (green) to MC (red) throughput for the "1P3S" allocation. From 5Pi QoS.

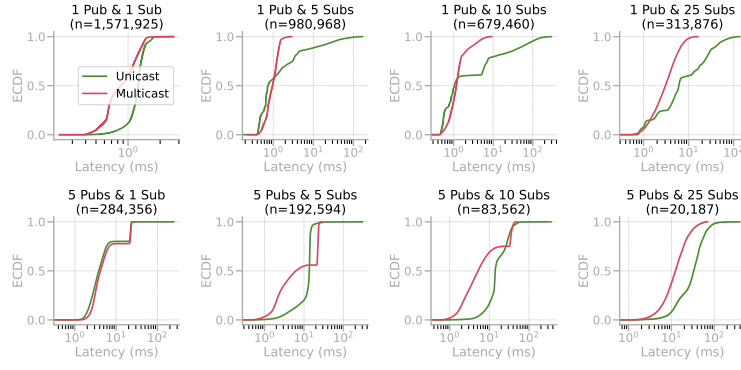


Figure B.22: eCDFs comparing UC (green) to MC (red) latency for the “2P2S” allocation. From 5Pi QoS.

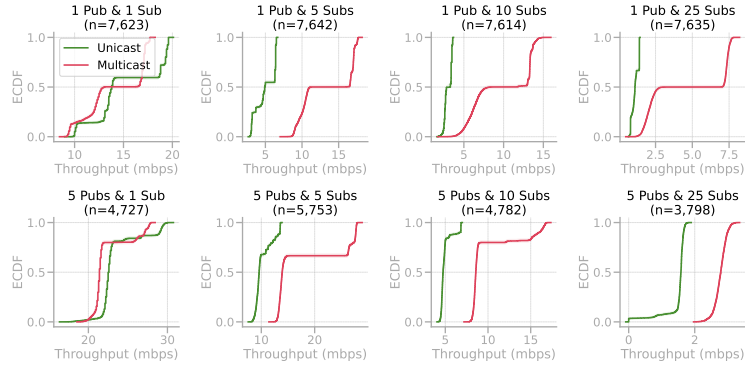


Figure B.23: eCDFs comparing UC (green) to MC (red) throughput for the “2P2S” allocation. From 5Pi QoS.

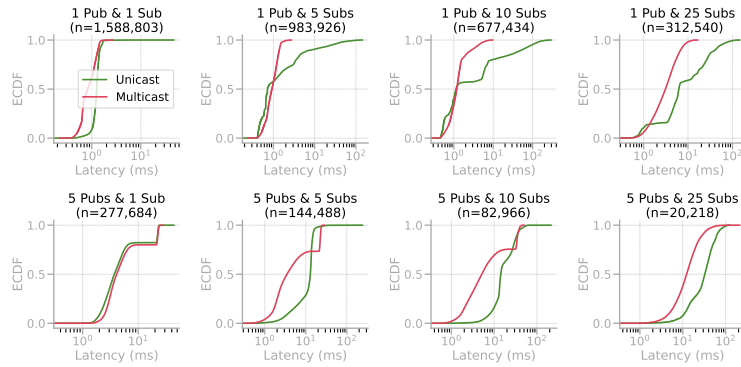


Figure B.24: eCDFs comparing UC (green) to MC (red) latency for the “3P1S” allocation. From 5Pi QoS.

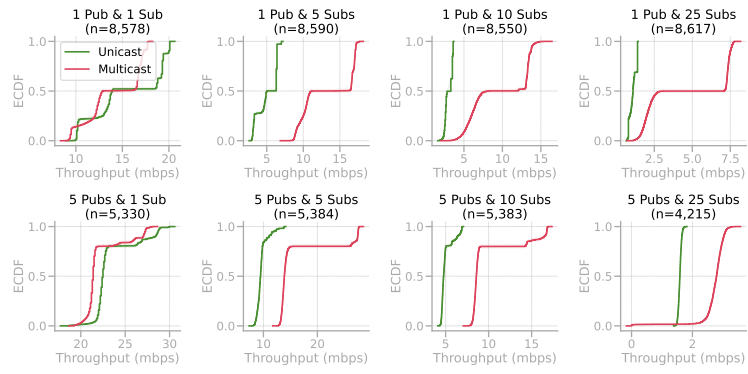
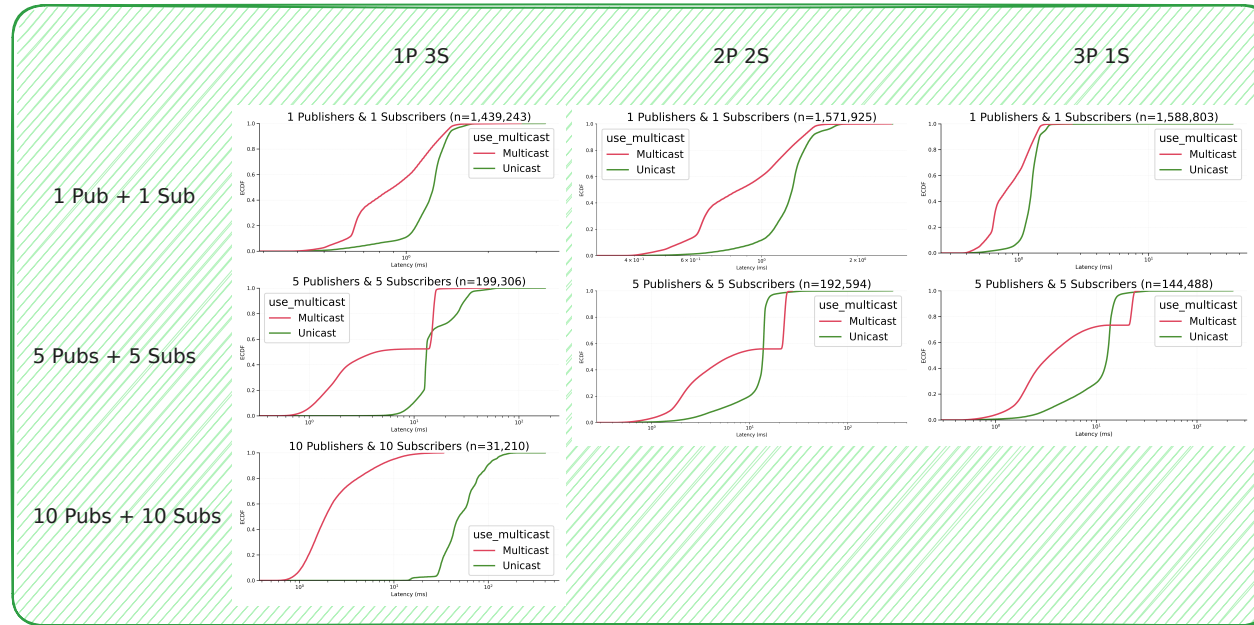


Figure B.25: eCDFs comparing UC (green) to MC (red) throughput for the "3P1S" allocation. From 5Pi QoS.

Latency



Throughput

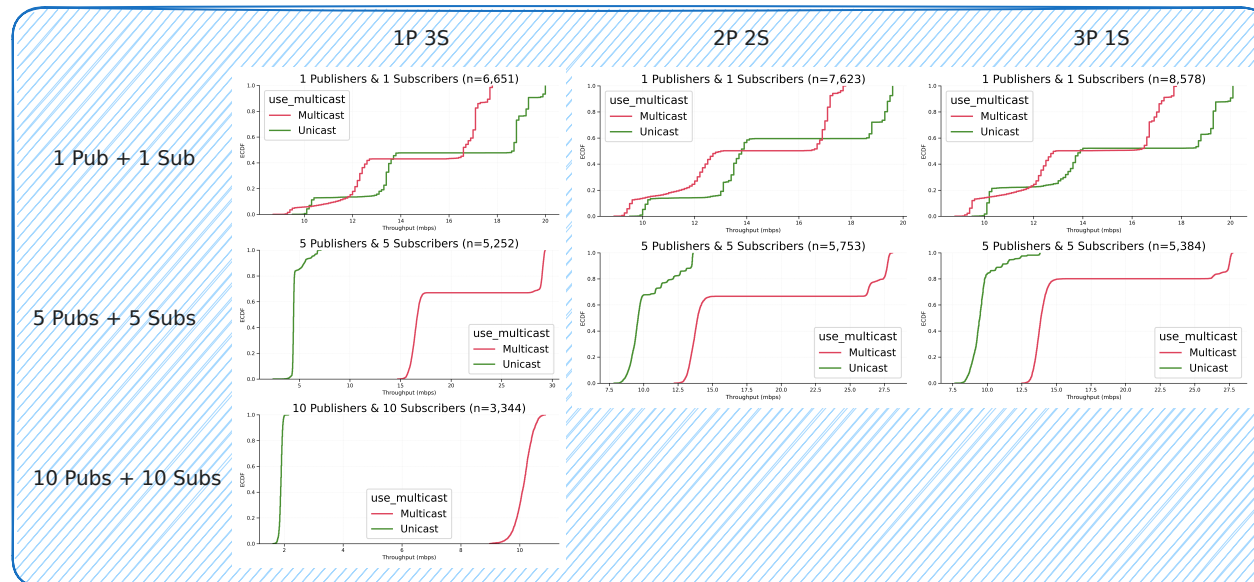


Figure B.26: Multicast analysis using the 5Pi QoS Capture Dataset looking at different participant allocations.

The experimental evaluation reveals distinct performance characteristics between UC and MC configurations across various publisher and subscriber counts. For latency, MC consistently outperforms UC in the configuration with one publisher and one subscriber. A consistent pattern is observed in the latency distribution for MC, represented by the red line, particularly in the scenario with five publishers and five subscribers. Notably, MC significantly outperforms UC in configurations with ten publishers and ten subscribers, as well as in the allocation of one publisher and three subscribers. Regarding throughput, MC consistently surpasses UC in scenarios involving more than one publisher and more than one subscriber. This performance advantage is particularly pronounced in the configurations with ten publishers and ten subscribers and one publisher with three subscribers.

These findings prompt further consideration of the underlying factors driving MC's performance. The consistent shape of the MC latency distribution (red line) indicates that hosting publishers and subscribers on separate machines has no significant impact on performance. Moreover, subscribers consistently receive a higher volume of data when using MC compared to UC.

These observations raise critical questions for future investigation. Why do subscribers receive more data when employing MC? Additionally, why are MC latencies lower than those of UC, particularly in the configuration with one publisher and one subscriber? Addressing these questions will enhance understanding of MC's efficiency and inform its application in data distribution systems.

1P3S

2P2S

3P1S

Latency

Throughput



Figure B.27: Multicast analysis using the 5Pi QoS Capture Dataset looking at all combinations of publishers and subscribers

B.2 Data Length Quantile Analysis

B.2.1 Latency

Overall

A summary of the latency analysis reveals several key findings. In the comparison between 100 B and 1 KB, the larger data length surprisingly outperforms the smaller one in 14 of 27 cases for the 95th to 100th percentile. The publisher count was identified as having a significant impact on this outcome. For the 100 B versus 512 B comparison, the 512 B data length was superior in 3 out of 10 cases in the highest percentile range, although no statistically significant parameter relationships were found. In the single case comparing 512 B and 1 KB, the 512 B data length was the clear winner. When comparing 100 B, 512 B, and 1 KB, the 512 B data length outperformed 100 B in 9 of 17 cases in the 5th to 95th percentile range, with publisher count again showing a strong relationship. Finally, in the comparison of a wide range of data lengths from 32 B to 1024 B, the 512 B data length was the most performant in one specific configuration, even outperforming smaller data lengths. The 128 B data length was optimal in 2 of 9 cases for the highest percentile, where the reliability setting was found to have a strong relationship with performance.

100 B

The 100 B data length demonstrated strong performance in several specific scenarios. It outperformed the 512 B data length in 7 out of 10 worst-case scenarios, particularly in configurations with higher publisher counts, lower subscriber counts, reliable communication, and UC. When compared against both 512 B and 1 KB, the 100 B data length was superior in 6 of 17 worst-case scenarios, typically with medium to high publisher and subscriber counts, reliable UC, and zero durability. In the 5th to 95th percentile range, it outperformed both 512 B and 1 KB in one case, with 5 publishers, 10 subscribers, reliable MC, and zero durability. Against the 1 KB data length alone, the 100 B option was better in 4 of 27 worst-case scenarios, with 10 publishers, reliable communication, and a mix of UC and MC. Furthermore, it was superior in 10 of 27 cases in the 5th to 95th percentile range, in configurations with a peak at 10 publishers, lower subscriber counts, and a preference for best-effort MC.

Contenders	Range	Cases	Pubs	Subs	Reliability	Multicast	Durability
512 B	95 to 100	7 of 10	Higher	Lower	reliable	UC	0
512 B, 1 KB	95 to 100	6 of 17	Mid to High	Mid	reliable	UC	0
1 KB	95 to 100	4 of 27	Mid	Spread	reliable	UC/multicast	0
1 KB	5 to 95	10 of 27	Low to Mid	Lower	best effort	MC	0

Table B.1: Summary of 100 B performance against other data lengths.

512 B

The 512 B data length also showed notable performance advantages in specific contexts. It outperformed the 100 B data length in 3 out of 10 cases in the 95th to 100th percentile range, particularly in configurations with lower publisher and subscriber counts, best-effort reliability, and MC. When compared against both 100 B and 1 KB, the 512 B option was superior in 3 of 17 cases in the 5th to 95th percentile range, favouring lower publisher counts, low to medium subscriber counts, and reliable UC. In the highest percentile range, it outperformed both 100 B and 1 KB in 9 of 17 cases, typically with lower publisher counts and 5 subscribers, using reliable communication. Furthermore, in a wide-ranging comparison, the 512 B data length beat all other options from 32 B to 1024 B in one worst-case scenario, which was configured with a single publisher, 3 subscribers, and reliable UC.

Contenders	Range	Cases	Pubs	Subs	Reliability	Multicast	Durability
100 B	95 to 100	3 of 10	Low	Low	best effort	MC	0
100 B, 1 KB	5 to 95	3 of 17	Low	Low to mid	reliable	UC	0
100 B, 1 KB	95 to 100	9 of 17	Low	Low	reliable	5 UC vs 4 MC	0
32 B, 64 B, 128 B, 256 B, 1024 B	95 to 100	1 of 9	1	Low	reliable	UC	0

Table B.2: Summary of 512 B performance against other data lengths.

Conclusion

The latency analysis compared data lengths ranging from 32 B to 1024 B. As anticipated, the performance differences between the various data lengths were not substantial, with most comparisons resulting in intersecting performance curves. However, some unexpected results emerged. The 100 B data length demonstrated superior performance in the worst-case scenarios (95th to 100th percentile) when compared to 512 B and 1 KB, particularly with a wide range of publisher counts, low to medium subscriber counts, and a mix of reliability and communication protocols. This suggests that while 100 B generally performs similarly to other data lengths below 1 KB, its advantage is most pronounced in these worst-case conditions, especially with lower subscriber counts. The 512 B data length also showed distinct advantages. It outperformed both 100 B and 1 KB in 3 of 17 cases in the 5th to 95th percentile range, particularly with low publisher and subscriber counts and reliable UC. Furthermore, it proved to be the most performant option in the highest percentile range when compared against all other data lengths, under specific conditions of low publisher and subscriber counts with reliable communication. Therefore, for a system constrained to low publisher and subscriber counts with a reliable protocol, the 512 B data length is the recommended choice for optimal latency performance.

B.2.2 Throughput

Parameter	0 to 5 χ^2	0 to 5 P- Value	0 to 5 DoF	5 to 95 χ^2	5 to 95 P- Value	5 to 95 DoF	95 to 100 χ^2	95 to 100 P- Value	95 to 100 DoF	0 to 100 χ^2	0 to 100 P- Value	0 to 100 DoF
Publisher Count	4.44444	0.2173	3	5.33333	0.148954	3	4.44444	0.2173	3	4.44444	0.2173	3
Subscriber Count	1.66667	0.64437	3	8	0.0460117	3	2.59259	0.45879	3	1.66667	0.64437	3
Reliability	0	1	0	0	1	0	0	1	0	0	1	0
Multicast	0	1	1	0	1	1	0	1	1	0	1	1
Durability	0	1	0	0	1	0	0	1	0	0	1	0

Table B.3: χ^2 correlation analysis for throughput between 100 B and 512 B.

Parameter	0 to 5 Cramers V	5 to 95 Cramers V	95 to 100 Cramers V	0 to 100 Cramers V
Publisher Count	0.666667	0.730297	0.666667	0.666667
Subscriber Count	0.408248	0.894427	0.509175	0.408248
Reliability	0	0	0	0
Multicast	0	0	0	0
Durability	0	0	0	0

Table B.4: V correlation analysis for throughput between 100 B and 512 B.

512 B vs 1 KB

	0 to 5 Count	5 to 95 Count	95 to 100 Count	0 to 100 Count
512 B	0	1	1	0
Uncategorised	1	0	0	1
Total	1	1	1	1

Table B.5: Counts of best performing data lengths for throughput between 512 B and 1 KB.

Distribution Analysis

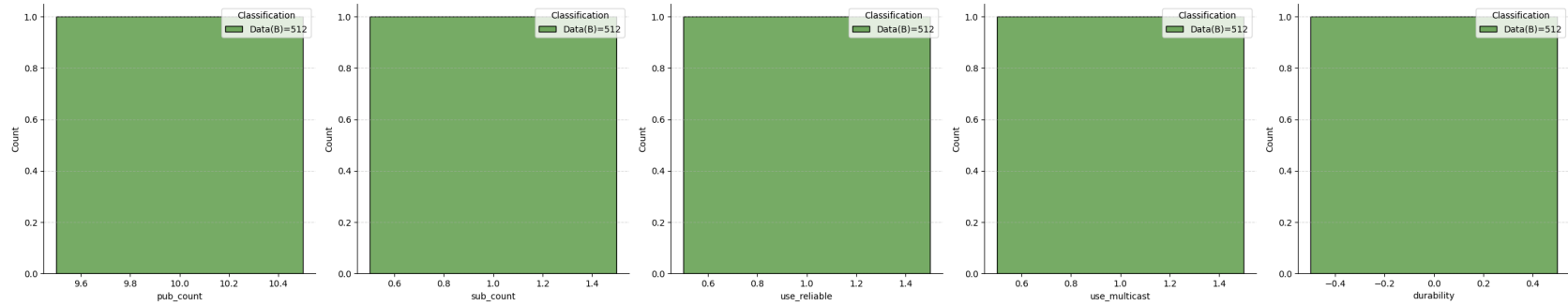


Figure B.28: Histogram showing distribution of best performing data lengths per configuration parameter between 5 and 95 percentiles for comparison of 512 B and 1 KB.

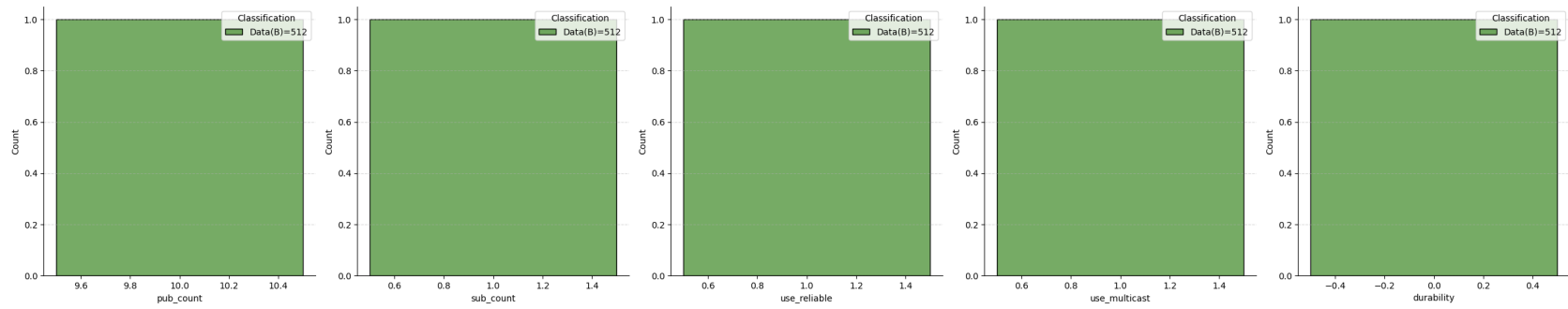


Figure B.29: Histogram showing distribution of best performing data lengths per configuration parameter between 95 and 100 percentiles for comparison of 512 B and 1 KB.

In the 5th to 95th percentile range, the 512 B data length was optimal in a single configuration of 10 publishers, 1 subscriber, reliable MC, and zero durability. Similarly, in the 95th to 100th percentile range, the 512 B data length was again superior in the same configuration of 10 publishers, 1 subscriber, reliable MC, and zero durability.

Parameter	0 to 5 χ^2	0 to 5 P- Value	0 to 5 DoF	5 to 95 χ^2	5 to 95 P- Value	5 to 95 DoF	95 to 100 χ^2	95 to 100 P- Value	95 to 100 DoF	0 to 100 χ^2	0 to 100 P- Value	0 to 100 DoF
Publisher Count	0	1	0	0	1	0	0	1	0	0	1	0
Subscriber Count	0	1	0	0	1	0	0	1	0	0	1	0
Reliability	0	1	0	0	1	0	0	1	0	0	1	0
Multicast	0	1	0	0	1	0	0	1	0	0	1	0
Durability	0	1	0	0	1	0	0	1	0	0	1	0

Table B.6: χ^2 correlation analysis for throughput between 512 B and 1 KB.

Parameter	0 to 5 Cramers V	5 to 95 Cramers V	95 to 100 Cramers V	0 to 100 Cramers V
Publisher Count	0	0	0	0
Subscriber Count	0	0	0	0
Reliability	0	0	0	0
Multicast	0	0	0	0
Durability	0	0	0	0

Table B.7: V correlation analysis for throughput between 512 B and 1 KB.

Statistical Analysis

Bibliography

Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. (2019), Optuna: A Next-generation Hyperparameter Optimization Framework, *in* ‘Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining’, KDD ’19, Association for Computing Machinery, New York, NY, USA, pp. 2623–2631.

URL: <https://doi.org/10.1145/3292500.3330701>

Al-Madani, B., Al-Roubaiey, A. & Baig, Z. A. (2014), ‘Real-time qos-aware video streaming: A comparative and experimental study’, *Advances in Multimedia* **2014**.

Al-Madani, B. & Ali, H. (2017), ‘Data distribution service (dds) based implementation of smart grid devices using ansi c12.19 standard’, *Procedia Computer Science* **110**, 394–401.

Almadani, B. (2016), ‘QoS-aware real-time pub/sub middleware for drilling data management in petroleum industry’, *Journal of Ambient Intelligence and Humanized Computing* **7**(2), 287–299.

Almadani, B., Aliyu, F. M. & Shakshuki, E. M. (2015), ‘Al-yad: A wearable sensor network over dds middleware for industrial application’, *Procedia Computer Science* **56**(1), 333–340.

Almadani, B., Bajwa, M. N., Yang, S. H. & Saif, A. W. A. (2015), ‘Performance evaluation of dds-based middleware over wireless channel for reconfigurable manufacturing systems’, *International Journal of Distributed Sensor Networks* **2015**.

URL: <https://doi.org/10.1155/2015/863123>

Amel, B. N., Rim, B., Houda, J., Salem, H. & Khaled, J. (2015), ‘Data distribution service on top of ethernet networks’, *2015 International Symposium on Networks, Computers and Communications, ISNCC 2015* pp. 1–5.

AMQP (n.d.).

URL: <https://www.amqp.org/>

- Andrei, G., Marlen, B., Sergey, T. & , K. (2020), ‘Industrial messaging middleware: Standards and performance evaluation’, *14th IEEE International Conference on Application of Information and Communication Technologies, AICT 2020 - Proceedings* .
- Balasubramanian, S., Ghosal, D., Sharath, K. N. B., Pouyoul, E., Sim, A., Wu, K. & Tierney, B. (2018), ‘Auto-tuned publisher in a pub/sub system: Design and performance evaluation’, *Proceedings - 15th IEEE International Conference on Autonomic Computing, ICAC 2018* pp. 21–30.
- Bao, L., Liu, X., Xin Liu, Xin Liu, Ziheng, X. & Fang, B. (2018), ‘AutoConfig: automatic configuration tuning for distributed message systems’, pp. 29–40. MAG ID: 2888705583.
- Basanta-val, P. & Serrano-torres, R. (n.d.), ‘Benchmarking communication middleware for cloud computing virtualizers’, *2nd International Workshop on Real-Time and Distributed Computing in Emerging Applications* .
- BATCH QosPolicy (DDS Extension)* (n.d.).
- URL:** https://community.rti.com/static/documentation/connect-dds/5.2.3/doc/manuals/connect_dds/html_files/RTI-ConnectDDS_CoreLibraries_UsersManual/Content/UsersManual/BATCH_Qos.htm#sending_2410472787_2949205
- Bellavista, P., Corradi, A., Foschini, L. & Pernafini, A. (2013), ‘Data distribution service (dds): A performance comparison of opensplice and rti implementations’, *Proceedings - International Symposium on Computers and Communications* pp. 377–383.
- Bouloukakis, G., Kattepur, A., Georgantas, N. & Issarny, V. (2018), ‘Queueing Network Modeling Patterns for Reliable and Unreliable Publish/Subscribe Protocols’, pp. 176–186. MAG ID: 2904555933.
- Breiman, L. (2001), ‘Random Forests’, *Machine Learning* **45**(1), 5–32.
- URL:** <https://doi.org/10.1023/A:1010933404324>
- Chen, T. & Guestrin, C. (2016), XGBoost: A Scalable Tree Boosting System, in ‘Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining’, pp. 785–794. arXiv:1603.02754 [cs].
- URL:** <http://arxiv.org/abs/1603.02754>
- Chen, Y. & Kunz, T. (2016), ‘Performance evaluation of iot protocols under a constrained

wireless access network’, *2016 International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT 2016* pp. 1–7.

Data Distribution Service (DDS) (n.d.).

Data Distribution Service Specification Version 2.5 (n.d.). Page 139.

URL: <https://www.omg.org/spec/DDS-RTSP/2.5>

Dinno, A. (2015), ‘Nonparametric Pairwise Multiple Comparisons in Independent Groups using Dunn’s Test’, *The Stata Journal: Promoting communications on statistics and Stata* **15**(1), 292–300.

URL: <https://journals.sagepub.com/doi/10.1177/1536867X1501500117>

Do, Q. & Franklin, S. (2011), ‘An investigation on the applicability of dds middleware as a systems integration tool’, *21st Annual International Symposium of the International Council on Systems Engineering, INCOSE 2011* **2**(Nandagopal 2006), 1177–1188.

ecdf — SciPy v1.16.2 Manual (n.d.).

URL: <https://docs.scipy.org/doc/scipy-1.16.2/reference/generated/scipy.stats.ecdf.html>

eProxima Fast DDS (n.d.).

URL: <https://www.eprosima.com/index.php/products-all/eprosima-fast-dds>

Esposito, C., Russo, S. & Di Crescenzo, D. (2008), ‘Performance assessment of omg compliant data distribution middleware’, *IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM*.

Essers, M. S. & Vaneker, T. H. (2014), ‘Evaluating a prototype approach to validating a dds-based system architecture for automated manufacturing environments’, *Procedia CIRP* **25**(C), 385–392.

Ferreira, R. D. F. & De Oliveira, R. S. (2017), ‘Cloud iec 61850: Dds performance in virtualized environment with opendds’, *IEEE CIT 2017 - 17th IEEE International Conference on Computer and Information Technology* pp. 231–236.

Fujdiak, R., Misurec, J., Pokorný, J., Zobal, L., Popov, P., Stankovic, V., Mlynek, P., Mrnustik, P., Blazek, P. & Musil, P. (2019), ‘Security and performance trade-offs for data distribution service in flying ad-hoc networks’, *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops* **2019-Octob.**

- García-Valls, M. & Basanta-Val, P. (2017), ‘Analyzing point-to-point dds communication over desktop virtualization software’, *Computer Standards and Interfaces* **49**, 11–21.
- Gauss, C. F. (n.d.), *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*, Little, Brown and Company, Boston.
- Google Scholar (n.d.).
URL: <https://scholar.google.com/>
- Grebhahn, A., Siegmund, N. & Apel, S. (2019), ‘Predicting Performance of Software Configurations: There is no Silver Bullet.’, *arXiv: Software Engineering* . MAG ID: 2991131477.
- Guo, J., Yang, D., Siegmund, N., Apel, S., Sarkar, A., Valov, P., Czarnecki, K., Wasowski, A., Huiqun Yu, Huiqun Yu & Yu, H. (2018), ‘Data-efficient performance learning for configurable systems’, *Empirical Software Engineering* **23**(3), 1826–1867. MAG ID: 2769879317.
- Ha, H., Ha, H. & Zhang, H. (2019a), ‘DeepPerf: performance prediction for configurable software with deep sparse neural network’, pp. 1095–1106. MAG ID: 2954141573.
- Ha, H., Ha, H. & Zhang, H. (2019b), ‘Performance-Influence Model for Highly Configurable Software with Fourier Learning and Lasso Regression’, pp. 470–480. MAG ID: 2992673542.
- Hakiri, A., Berthou, P. & Gayraud, T. (2010), ‘Addressing the challenge of distributed interactive simulation with data distribution service’, *2010 International Simulation Multi-Conference* pp. 103–111.
- High Level Architecture* (n.d.), NATO.
- Innovations, R.-T. (n.d.), ‘Connex Product Suite for Intelligent Distributed Systems’.
URL: <https://www.rti.com/products>
- Kang, Z. & Canady, R. (n.d.), ‘A study of publish / subscribe middleware under different iot traffic conditions’, *Proceedings of the International Workshop on Middleware and Applications for the Internet of Things* .
- Kavimandan, A. & Gokhale, A. (2007), ‘Automated middleware qos configuration techniques using model transformations’, *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC* pp. 20–27.
- Kenney, J. F. & Keeping, E. S. (n.d.), Mathematics of Statistics Part One, in ‘Mathematics of Statistics’, 3rd edn, pp. 252–285.

- Kim, C. H., Yoon, G., Lee, W., Park, J. & Choi, H. (2015), ‘A performance simulator for dds networks’, *International Conference on Information Networking* **2015-Janua**, 122–126.
- Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J. & Linkman, S. (2009), ‘Systematic literature reviews in software engineering—a systematic literature review’, *Information and software technology* **51**(1), 7–15.
- Köksal, Ö. & Tekinerdogan, B. (2017), ‘Obstacles in data distribution service middleware: A systematic review’, *Future Generation Computer Systems* **68**, 191–210.
URL: <http://dx.doi.org/10.1016/j.future.2016.09.020>
- Krinkin, K., Filatov, A., Filatov, A., Kurishev, O. & Lyanguzov, A. (2018), Data distribution services performance evaluation framework, in ‘2018 22nd Conference of Open Innovations Association (FRUCT)’, IEEE, pp. 94–100.
- Kruskal, W. H. & Wallis, W. A. (1952), ‘Use of Ranks in One-Criterion Variance Analysis’, *Journal of the American Statistical Association* **47**(260), 583–621. Publisher: ASA Website
 eprint: <https://www.tandfonline.com/doi/pdf/10.1080/01621459.1952.10483441>.
URL: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1952.10483441>
- Lamport, L. (1978), ‘Time, clocks, and the ordering of events in a distributed system’, *Communications of the ACM* **21**(7), 558–565.
URL: <https://dl.acm.org/doi/10.1145/359545.359563>
- Lee, K. H., Noh, T. W., Kim, K. T., Kim, W. T. & Kim, C. K. (2015), ‘Router design for dds: Architecture and performance evaluation’, *Applied Mathematics and Information Sciences* **9**(2), 513–521.
- Macenski, S., Foote, T., Gerkey, B., Lalancette, C. & Woodall, W. (2022), ‘Robot operating system 2: Design, architecture, and uses in the wild’, *Science Robotics* **7**(66).
URL: <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>
- Mann, H. B. & Whitney, D. R. (1947), ‘On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other’, *The Annals of Mathematical Statistics* **18**(1), 50–60. Publisher: Institute of Mathematical Statistics.
URL: <https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-18/issue-1/On-a-Test-of-Whether-one-of-Two-Random-Variables/10.1214/aoms/1177730491.full>

Maruyama, Y., Kato, S. & Azumi, T. (2016), ‘Exploring the performance of ros2’, *Proceedings of the 13th International Conference on Embedded Software, EMSOFT 2016* pp. 0–9.

MilSOFT (n.d.).

URL: <https://www.milsoft.com.tr/index.php/portfolio/mildds-en/>

MQTT (n.d.).

URL: <https://mqtt.org/>

Napolitano, A., Carrozza, G., Bovenzi, A. & Esposito, C. (2011), ‘Automatic robustness assessment of dds-compliant middleware’, *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, PRDC* pp. 302–307.

Object Management Group (n.d.).

URL: <https://www.omg.org/>

OpenDDS (n.d.).

URL: <https://opendds.org/>

pandas.DataFrame (n.d.).

URL: <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>

pandas - Python Data Analysis Library (n.d.).

URL: <https://pandas.pydata.org/>

Parquet (n.d.).

URL: <https://parquet.apache.org/>

Peeroo, K., Popov, P. & Stankovic, V. (2022), ‘Exploring the Effects of Multicast Communication on DDS Performance’. Version Number: 1.

URL: <https://arxiv.org/abs/2209.09001>

Peeroo, K., Popov, P., Stankovic, V. & Weyde, T. (2024), Machine Learning for Performance Prediction of Data Distribution Service (DDS), in ‘2024 19th European Dependable Computing Conference (EDCC)’, IEEE, Leuven, Belgium, pp. 111–114.

URL: <https://ieeexplore.ieee.org/document/10533301/>

Profanter, S., Tekat, A., Dorofeev, K., Rickert, M. & Knoll, A. (2019), ‘Opc ua versus ros, dds, and mqtt: Performance evaluation of industry 4.0 protocols’, *Proceedings of the IEEE International Conference on Industrial Technology 2019-Febru*, 955–962.

- Rathanasalam, R. & Kanagasabai, K. (2020), ‘Data dissemination with interoperability in iot network’, *International Journal of Communication Systems* **33**(15), 1–17.
- Rausch, M. & Sanders, W. H. (2020), Sensitivity Analysis and Uncertainty Quantification of State -Based Discrete-Event Simulation Models Through a Stacked Ensemble of Metamodels, *in* M. Gribaudo, D. N. Jansen & A. Remke, eds, ‘Quantitative Evaluation of Systems’, Vol. 12289, Springer International Publishing, Cham, pp. 276–293. Series Title: Lecture Notes in Computer Science.
URL: http://link.springer.com/10.1007/978-3-030-59854-9_20
- Rhoades, A. S., Schrader, G. & Poulin, P. (2007), ‘Benchmarking publish / subscribe middleware for radar applications communications middleware benchmarking application’, (May), 1–2.
- RTI* (n.d.).
URL: <https://www.rti.com/en/>
- RTI Connex* (n.d.).
URL: <https://www.rti.com/products>
- RTI Connex DDS Core Libraries User’s Manual* (2016).
URL: www.ANTLR.org
- RTI PerfTest* (2023). original-date: 2016-08-23T20:27:26Z.
URL: <https://github.com/rticomunity/rtiperftest>
- RTI PerfTest* (n.d.).
URL: <https://github.com/rticomunity/rtiperftest>
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), ‘Learning representations by back-propagating errors’, *Nature* **323**, 533–536. ADS Bibcode: 1986Natur.323..533R.
URL: <https://ui.adsabs.harvard.edu/abs/1986Natur.323..533R>
- Sachs, K., Kounev, S. & Buchmann, A. (2013), ‘Performance modeling and analysis of message-oriented event-driven systems’, *Software & Systems Modeling* **12**(4), 705–729.
URL: <http://link.springer.com/10.1007/s10270-012-0228-1>
- Serrano-Torres, R., García-Valls, M. & Basanta-Val, P. (2013), ‘Performance evaluation of virtualized dds middleware’, *Simposio de tiempo real español*.

SHAPIRO, S. S. & WILK, M. B. (1965), ‘An analysis of variance test for normality (complete samples)[†]’, *Biometrika* **52**(3-4), 591–611.

URL: <https://doi.org/10.1093/biomet/52.3-4.591>

Shu, Y., Sui, Y., Hongyu Zhang, Zhang, H. & Xu, G. (2020), ‘Perf-AL: Performance Prediction for Configurable Software through Adversarial Learning’. MAG ID: 3094340071.

Siegmund, N., Grebhahn, A., Apel, S. & Kästner, C. (2015), Performance-influence models for highly configurable systems, *in* ‘Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering’, ACM, Bergamo Italy, pp. 284–294.

URL: <https://dl.acm.org/doi/10.1145/2786805.2786845>

sklearn.preprocessing.MinMaxScaler (n.d.).

URL: <https://scikit-learn/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

sklearn.preprocessing.RobustScaler (n.d.).

URL: <https://scikit-learn/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>

sklearn.preprocessing.StandardScaler (n.d.).

URL: <https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

Sthle, L. & Wold, S. (1989), ‘Analysis of variance (ANOVA)’, *Chemometrics and Intelligent Laboratory Systems* **6**(4), 259–272.

URL: <https://www.sciencedirect.com/science/article/pii/0169743989800954>

Thales Group (n.d.).

URL: <https://www.thalesgroup.com/en>

Tukey, J. W. (1949), ‘Comparing Individual Means in the Analysis of Variance’, *Biometrics* **5**(2), 99–114. Publisher: [Wiley, International Biometric Society].

URL: <https://www.jstor.org/stable/3001913>

Turkay, E., Gokhale, A. & Natarajan, B. (2004), ‘Addressing the middleware configuration challenges using model-based techniques’, *Proceedings of the Annual Southeast Conference* pp. 166–170.

Vortex OpenSplice (n.d.).

URL: <https://www.adlinktech.com/en/vortex-opensplice-data-distribution-service>

WELCH, B. L. (1947), ‘THE GENERALIZATION OF ‘STUDENT’S’ PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARLANCES ARE INVOLVED’, *Biometrika* **34**(1-2), 28–35.

URL: <https://doi.org/10.1093/biomet/34.1-2.28>

Xiong, M., Parsons, J., Edmondson, J., Nguyen, H. & Shmidt, D. C. (2010), ‘Evaluating the performance of publish/subscribe platforms for information management in distributed real-time and embedded systems’, *OMG Whitepapers: Data Distribution Service Portal* .

URL: http://portals.omg.org/dds/sites/default/files/Evaluating_Performance_Publish_Subscribe_Platforms.pdf

Yang, J., Sandstrom, K., Nolte, T. & Behnam, M. (2012), ‘Data distribution service for industrial automation’, *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* pp. 1–8.

Youssef, T. A., Elsayed, A. T. & Mohammed, O. A. (2015), Dds based interoperability framework for smart grid testbed infrastructure, in ‘2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)’, IEEE, pp. 219–224.

Youssef, T. A., Hariri, M. E., Elsayed, A. T. & Mohammed, O. A. (2018), ‘A dds-based energy management framework for small microgrid operation and control’, *IEEE Transactions on Industrial Informatics* **14**(3), 958–968.

ZeroMQ (n.d.).

URL: <https://zeromq.org/>

Zhang, Y., Guo, J., Blais, E. & Czarnecki, K. (2015), Performance Prediction of Configurable Software Systems by Fourier Learning (T), in ‘2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)’, IEEE, Lincoln, NE, USA, pp. 365–373.

URL: <http://ieeexplore.ieee.org/document/7372025/>