Tampereen ammattikorkeakoulu

MQTT Suitability for Performance-Critical Resource Control Communication

Nina Laaksonen

BACHELOR'S THESIS
December 2025

Degree Programme in ICT Engineering
Software Engineering

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Ohjelmistotekniikka

LAAKSONEN, NINA:
MQTT:n sopivuus suorituskykykriittiseen resurssinhallintakommunikaatioon

Opinnäytetyö 47 sivua, joista liitteitä 13 sivua
Joulukuu 2025

Opinnäytetyössä tutkittiin ja perusteltiin, kuinka MQTT-protokolla sopii kommunikointiin suorituskykyä vaativaan testiympäristöön. Opinnäytetyö tehtiin yhteistyössä kansainvälisen yrityksen kanssa. Työn tavoitteena oli tutkia ja testata, kuinka vahvasti MQTT-asiakkaiden viestien välinen nopeus eri MQTT-välittäjien ja viestien laatutasojen ja hyötykuorman kokojen ja formaattien yhdistelmä vaikuttaa nopeasti lähetettyjen viestien matkustusaikaan MQTT-asiakkaiden välillä. Tutkimuksessa tutkittiin välittäjiä EMQX, HiveMQ, NanoMQ ja RabbitMQ, ja siinä käytetyt testit suoritettiin konteissa olevien asiakkaiden avulla, jolloin saatiin jäljiteltyä todellisia testiympäristön olosuhteita ja muita yritykseltä tulevia vaatimuksia. Mittaukset suoritettiin itsenäisissä konteissa olevan kahden eri asiakkaan välisen kommunikoinnin mittaukseen ja kahden eri asiakkaan ollessa eri prosesseissa samassa kontissa. Näin varmistettiin mittatulosten yhteensopivuus ja vertailtavuus ja kellojen synkronisuus niin yhdensuuntaisen matkan kuin edestakaisen matkan aikojen mittaukseen. Opinnäytetyö kirjoitettiin englannin kielellä.

Mittauksien mittayksikkö oli mikrosekunnit, ja niillä mitattiin asiakkaiden välistä yhdensuuntaisen matkan, asiakkaan lähettämän viestin edestakaiseen matkaan kuluvaa aikaa, ja viestin työkuorman sarjoittamiseen ja sen käyttöön kulutettua aikaa. Jokaisesta yrityksen vaatimuksen luomasta yhdistelmästä suoritettiin useita eri mittauksia ja jokaiset yhdistelmät visualisoitiin erilaisin kuvaajin ja analysoitiin mahdolliset nousseet huomiot. Tulokset olivat odotusten mukaiset: viestien koko, vaatimustaso ja välittäjä vaikuttivat mitattuihin matkustusaikoihin, ja viestin työkuorman formaatilla oli yhteys sarjoittamiseen menevään aikaan. Kun otetaan huomioon mittayksikön tarkkuus, tietyillä viestintään käytetyillä asetusyhdistelmillä nähtävä erotus oli erittäin nimellinen, jos ei jopa mitätön. Merkittävimmät löydökset olivat viestien hyötykuorman koon ja laatutason ja välittäjien eroissa.

Tutkimuksen johtopäätös oli, että MQTT pystyy suoriutumaan annetuista vaatimuksista. Tuloksia tulkitessa suositusta oikeasta yhdistelmästä ei löytynyt vaan mahdollisia lisätutkimuksia on tehtävä. Aiheita ovat; miten käytetty laite tai verkon ja palomuurin yli kommunikointi vaikuttavat aikoihin, millainen vaikutus välittäjän asetusten vaihtamisella tai kuormittamisella eri asiakas- tai otsikkomääriin on ja vaikuttaako kommunikointiprotokollan vaihto viestien kuljetusaikoihin ja jos, mihin kannattaa vaihtaa.

Asiasanat: mqtt, testaus, kommunikointi, kontti, suorituskyky

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

LAAKSONEN, NINA:
MQTT Suitability for Performance-Critical Resource Control Communication

Bachelor's thesis 47 pages, appendices 13 pages
December 2025

The objective of this study was to test and gather information about MQTT performance suitability for controlling resources as the main communication protocol in between containerized services, test platform and resource control. The study was conducted in collaboration with a multinational company and is presented in perspective of the company demand to study and optimize communication. The theoretical section explored an overview of containers, MQTT and the test platform.

The data were collected in a realistic setup using custom-made tests for round-trip-time and single-trip-time. Different combinations of MQTT brokers, QoS levels, message payload sizes and formats were tested. The test results were visualized and analyzed systematically to answer the company requirements questions.

The results were within the requirements and a known phenomenon appeared: trip-times measured were variable with peaks, .NET compiling causes first run of tests last longer, different brokers, QoS and payload size caused notable differences in trip-times. These results suggest that MQTT meets the requirements. Further research is suggested to a few questions: how much do the running device and network with firewall effect, how much stressing the broker or optimizing broker configurations affect trip-times. In need of further optimization, it is suggested to research other communication protocols for this environment.

Key words: mqtt testing communication container performance

CONTENTS

TECHNICAL TERMS

AMQP	Advanced Message Queueing Protocol, protocol for transmitting messages
C#	programming language
DC	Direct Current, a type of electrical current that flows in one direction
.NET	dotnet, platform for building applications
container	standardised method of packaging and easily starting a software application
container image	read only template with working configurations, source code, libraries and variables for creating working container
gRPC	gRPC Remote Procedure Call, framework for efficient communication between systems
ICT	Information and Communication Technology
IoT	Internet of Things, a network for embedded ecosystem with sensors, software, and other technologies connected over the internet
Java	programming language
JIT	Just-In-Time, method of compiling code at runtime
JSON	JavaScript Object Notation, data format for structured data
KPI	Key Performance Indicator, type of performance measurement
MQTT	Message Queuing Telemetry Transport, communication protocol
MQTT broker	communication server handling the topics clients can publish and subscribe for and controlling message priorities and queues using standards like QoS
MQTT client	software communicating through subscribing or publishing MQTT broker topics

MQTT Explorer	desktop application with MQTT client subscribing to all topics on given broker to visualize and monitor the broker traffic
NuGet	.NET official package manager
protobuf	Protocol Buffers, method for serializing structured data
QoS	Quality of Service, mechanisms to ensure dependability and prioritize message delivery in a network
RTT	round-trip-time, the time message takes to travel from sender to receiver and back to sender
STT	single-trip-time, the time message takes to travel from sender to receiver
TAMK	Tampereen ammattikorkeakoulu, Tampere University of Applied Sciences
Typescript	programming language
UI	user interface

1 INTRODUCTION

This Bachelor's Thesis was conducted in collaboration with a Saab AB development team. Saab AB is a Swedish defence and security company with global presence that provides products, services, and solutions for military and civilian markets. Operations and customers are located across Europe, the Americas, Asia-Pacific and Africa. The research topic originated from the Saab AB's need to create a component for a flexible test platform that acts as a wrapper to control different resources on a test station. The original research topic focused on developing architecture for the resource-controlling software wrapper. However, while designing the architecture for the first minimum viable product (MVP) version, which targeted a simple use case involving a DC power supply unit, several additional research questions emerged. Upon consideration, it become evident that most of these questions were substantial enough for separate Bachelor's Thesis. This realisation led to re-evaluation of the scope and number of research questions at hand. It became clear the original research area needed to be re-scoped to focus on only one of the arisen research questions. The most suitable question for this research was: "is MQTT suitable for performance-critical resource control communication?"

The aim of this research is to investigate and evaluate the suitability of Message Queuing Telemetry Transport (MQTT) for resource control in a scalable test platform with intensive communication requirements between platform containers during resource controlling. Additionally, it aims to explore possible alternatives to MQTT in case it does not meet the requirements. Since the rest of the platform already uses MQTT for communication between containers, MQTT is the logical starting point for investigating its suitability between the resource control software wrapper and the test platform containers. The aim is to evaluate and compare the performance of several of the most suitable MQTT brokers using different message payload formats and sizes, as well as various Quality of Service (QoS) configurations within the platform, to ensure MQTT meets the requirements. If MQTT does not meet the requirements, the plan is to research possible alternative solutions. The outline of this thesis is to explain high-level information about the test platform and resource control, detail the communication requirements

from the perspective of MQTT, and present the results from the measurements and the resolution of the research.

There is wealth of benchmark research on various brokers in container environments using different MQTT message combinations. However, many of these studies employ diverse tools, programming languages, and network configurations for measurements. These variations can potentially impact the performance of rapid messaging. Multiple measurements show conflicting results, some exceeding and others meeting the company's requirement limits. Therefore, it is essential to ensure that the company's specific needs are met by conducting actual tests on a setup that mimics the intended deployment environment. (M. Jin 2023, D. Ansari 2023, EMQX Team (2) 2023.).

2 PREREQUISITE KNOWLEDGE

2.1 Containers

Containers are a popular platform for managing isolated environments, enabling applications to run in their own namespace. The isolation ensures that applications can operate independently of each other's and of the underlying system. Containers package configurations, software libraries, and code in a lightweight and portable manner. (Docker n.d.)

The most common way to use containers is to run them from a container image, which serves as a template for all libraries and dependencies used within a container. There are several official repositories for container images, such as the Docker Hub, which hosts collection of container images contributed by both official sources and the community. (Docker n.d.)

All of this is managed by container engine, a technology designed for building and containerizing applications. The container engine is responsible for creating, running, and managing containers based on the specific images, ensuring that they operate efficiently and securely within their isolated environment. (Docker n.d.)

2.2 MQTT

MQTT is messaging protocol that uses publish-subscribe pattern. It is most commonly used in Internet of Things (IoT) networking. MQTT brokers handle the mapping of the addresses using topics, which specify where messages are published or subscribed by MQTT clients. MQTT clients exchange messages by listening, subscribing to, as well as sending (publishing) messages on specific broker topics. The message structure includes the topic it should be published on, the payload (the actual message value), the QoS level, the mechanism by which the message should be acknowledged by the broker and other metadata not related to this research, such as packet ID, retain flag, and DUP flag. (EMQX Team 2025)

MQTT defines three QoS levels:

- QoS 0 (At Most Once): “fire and forget” without delivery guarantee.
- QoS 1 (At Least Once): ensures that the message is received at least once by using PUBACK acknowledgment from the broker.
- QoS 2 (Exactly Once): ensures that the message is delivered exactly once through a four-step handshake using PUBREC, PUBREL, and PUBCOMP packets.

These acknowledgement mechanisms are handled by the broker using a standardized message exchange process. (EMQX Team 2025.)

2.3 Test platform

The test platform is designed to have containerized services for different test platform functionalities for running, visualize and document the tests. Each of these services communicate with each other’s through MQTT broker by using custom-made interfaces designed for the specific need for the service itself. One of the test platform services is controlling resource controlling service.

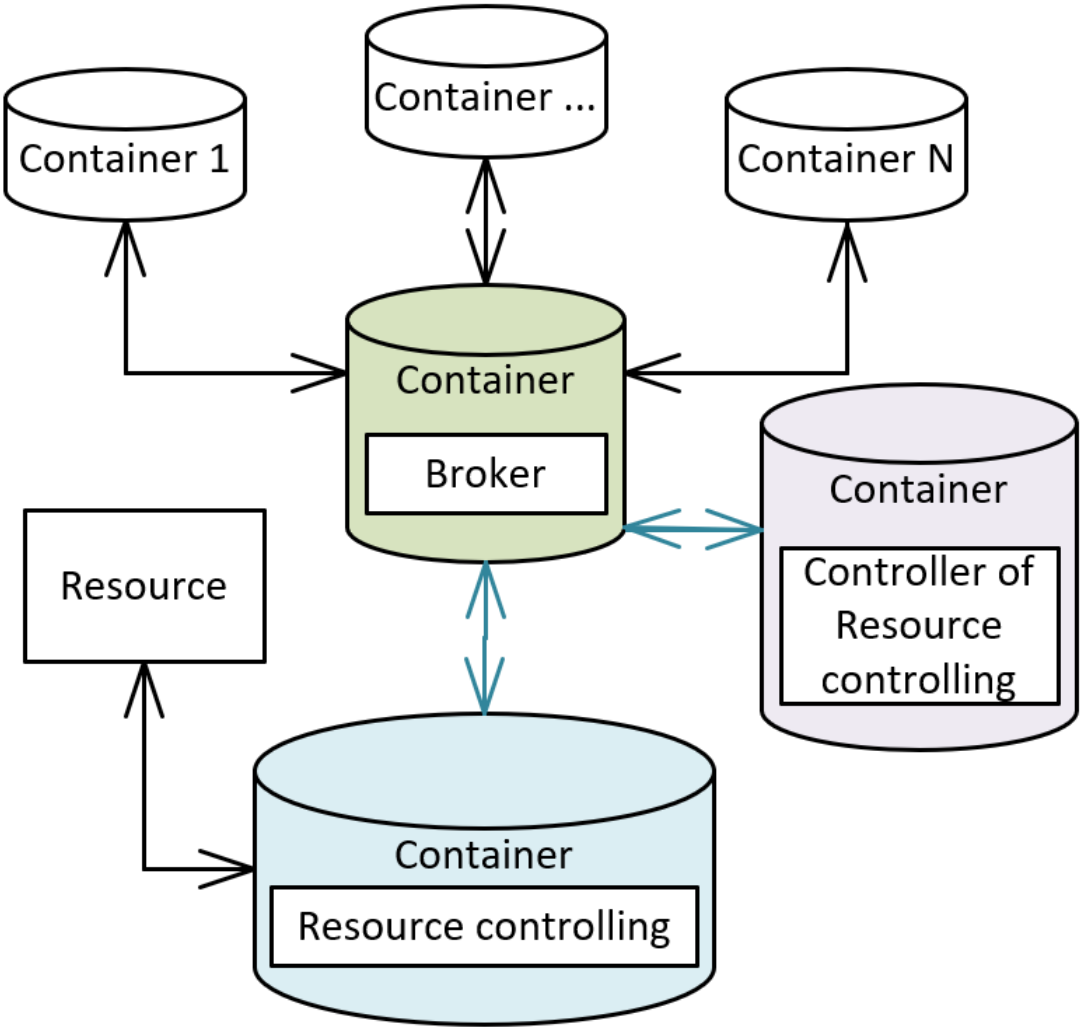


FIGURE 1. Test platform container structure.

This research main interest focuses on the communication between resource controlling service and the controller of resource controlling service, the blue lines seen in the FIGURE 1.

A few simple performance tests were conducted which showed that MQTT communication protocol on the unofficial setup was on unacceptable level. This discovery led to a requirement to do controlled performance testing between two MQTT clients in their individual container to mimic communication between services in charge of the resources.

3 MEASUREMENTS

3.1 Requirements from the company

The primary motivation for this research arose from drive to learn if MQTT can deliver the required performance for the resource controlling service. Some tests require thousands if not hundred thousand messages sent rapidly between resource and the resource controlling service and the platform design choices cannot be the reason for the test lasting week longer than it could. The development team will use drivers provided by the resource manufacturer, so control over the speed the resource controlling service communicates with the resource is out of development team hands. Therefore, the need for optimization lies between the controller of the resource controlling service and the resource controlling service as highlighted with blue arrows in the FIGURE 1.

All of the test platform services are designed to be in containers creating the need for MQTT clients used in this research to be in containers as well to mimic the real platform environment. Every message delivery needs to be confirmed which makes one of the measurements to be RTT. Standard research practice involves collecting all relevant information related the measurements, which creates the need for measure single-trip-time (STT). Both RTT and STT are explained in more detail under Measured data heading. After researching and communicating with different Saab site testing professionals about high number of messages rapidly sent in already made tests, it was decided the RTT should be equal or less than one millisecond. This is for keeping the already existing tests to last reasonable time while creating the need for measurement values to be on very accurate level, microseconds.

3.2 Hardware and software setup

Virtual machine setup for the measurements included:

- Operating system: Red Hat Enterprise Linux 9.6 (Plow)
- Processor: AMD 8-core
- Memory: 16G RAM
- Storage: 470 SSD

The measurement software was developed by using C#/.NET. Rootless Podman Compose was used to start the network environment, simulating the communication between the controller of the resource controlling service and the resource controlling service on the virtual machine. The following NuGet packages were utilized:

- MQTTnet: MQTT client controlling.
- System.Diagnostics.Process: Creating and managing different processes.
- NLog: Logging the measurements
- System.IO.MemoryMappedFiles: Transfer data between
- System.Diagnostics.Stopwatch: Measuring time on accurate enough level.

Log files generated during the measurements were processed using Python to create result files, namely 1-container-results.csv and 2-container-results.csv, which contains Key Performance Indicator (KPI) values. The Python library MatPlotLib was used for generating the charts and some visualizations were created with Microsoft Visio.

The following MQTT broker Docker images were obtained from Docker Hub:

- HiveMQ:
    - image: hivemq/hivemq-ce, version: 2022.1
- NanoMQ:
    - image: emqx/nanomq, version: 0.24.2
- RabbitMQ:
    - image: bitnami/rabbitmq, version: 4.1.2
- EMQX
    - image: emqx/emqx, version: 5.10.0

3.3 Data collection and measurements

The different measurements were made using all QoS levels to ensure comparability of the results and gather all relevant information from this research. It is important to note the measurement taken while using RabbitMQ broker do not include QoS 2, as RabbitMQ, as the RabbitMQ MQTT plugin does not support the actual definition of QoS 2 (Rabbit n.d.). Further information about RabbitMQ MQTT plugin functionalities can be found under the Conclusion section. For these measurements, each broker was taken in use with default Docker image configurations, with authentication activated.

The measurements were generated using brokers EMQX, HiveMQ, NanoMQ and RabbitMQ. NanoMQ and EMQX share the same base, with EMQX is designed to support clusters of brokers to create larger-scale, more available communication systems. In contrast, NanoMQ is stripped-down version of EMQX, optimized for light and rapid messaging without clustering feature (EMQX docs n.d.). Each of these brokers contain the qualities the company might be interested in now or in the future. HiveMQ was already in use on the actual test platform setup, making it natural choice to start the measurement and comparison. RabbitMQ was of interest to the company due to its various plugins, which offer a wide range of usage options.

Payload sizes were picked 0B, 50B, 1kB, 5kB, 10kB and 100kB. Initially, the goal was to measure large messages, up to 1MB, to ensure comprehensive coverage of message sizes. However, even though the largest package size supported for transmission using MQTT is 256MB, payload sizes larger than 100kB were excluded. This is because such sizes are more indicative of large file transfers rather than rapid communication, which is the focus of this research (EMQX Team (1) 2023).

The measurement log files were organised to facilitate easy access and analysis. Each combination of broker, QoS level, payload size, and format was measured five times, generating five sets of ten thousand measurement log files from the start of the container. These measurements were conducted in two setups: one

with a single container for clients and another with two containers for clients. Each setup produced corresponding log files. The data from these log files were compiled into two CSV files: 1-container-results.csv and 2-container-results.csv. The 2-container-results.csv file contains KPI values derived from RTT measurements, while the 1-container-results.csv file contains KPI values from STT and deserialization measurements. Each of the five runs contributes a line to the corresponding CSV file.

During the research several decisions were made about what data to gather and what data to use from all the gathered data. The most straightforward data to start with was to obtain the RTT from two different client in separate containers to mimic the exact design used in the actual environment. In this context, Client1 refers to the controller of the resource controlling service, while Client2 refers to the resource controlling service. The two-container setup RTT measurement is visualised in FIGURE 2 by the arrows and provides the time taken for both the blue and purple arrows to transfer the message.

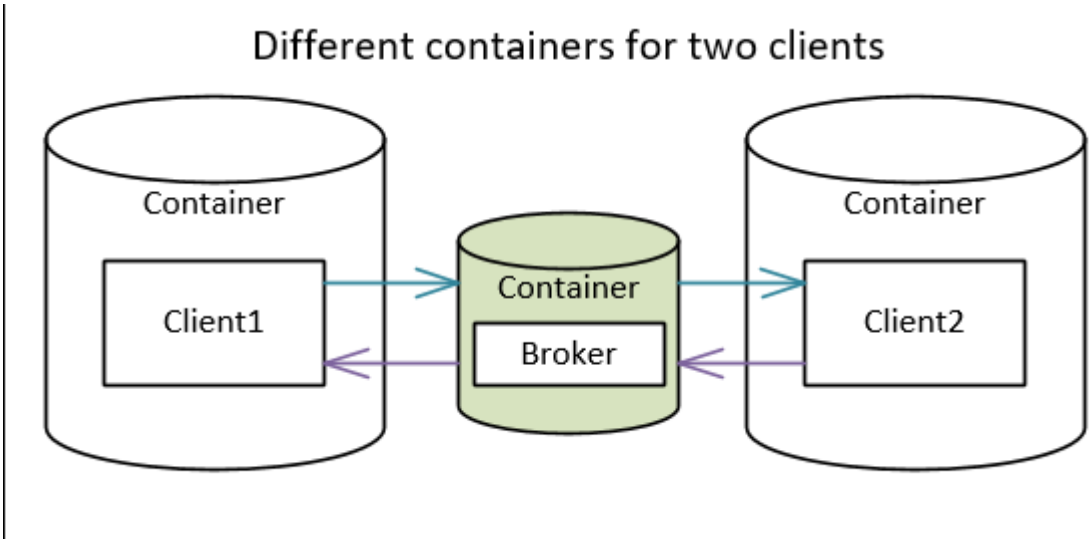


FIGURE 2. Communication among separate containers with individual MQTT clients and MQTT broker.

To achieve sufficient accuracy for STT and message deserializing time measurements, an issue arose regarding the system clock between two containers was not being synchronized. The most straightforward approach to address this issue was to set two MQTT clients in one container but run them in different system processes, as demonstrated in FIGURE 3.

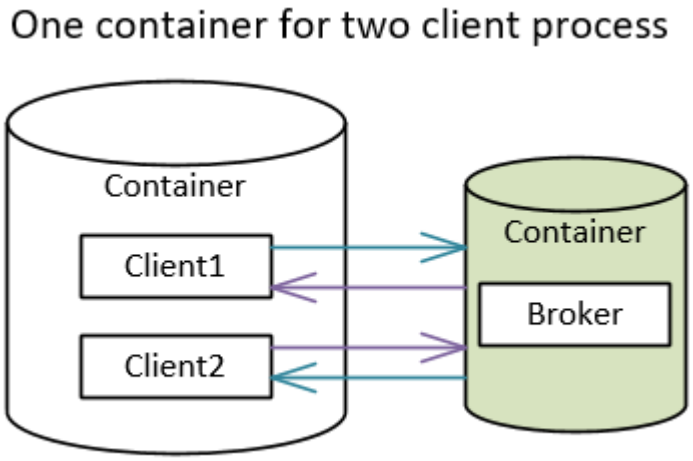


FIGURE 3. Communication between two MQTT clients in one container and another container for MQTT broker.

The first STT is the time measured from the exact moment Client1 sends the message to Client2 to when Client2 receives it, represented by the blue arrows on FIGURE 3. Similarly, the second STT is measured from Client2 to Client1, represented by the purple arrows on the same figure.

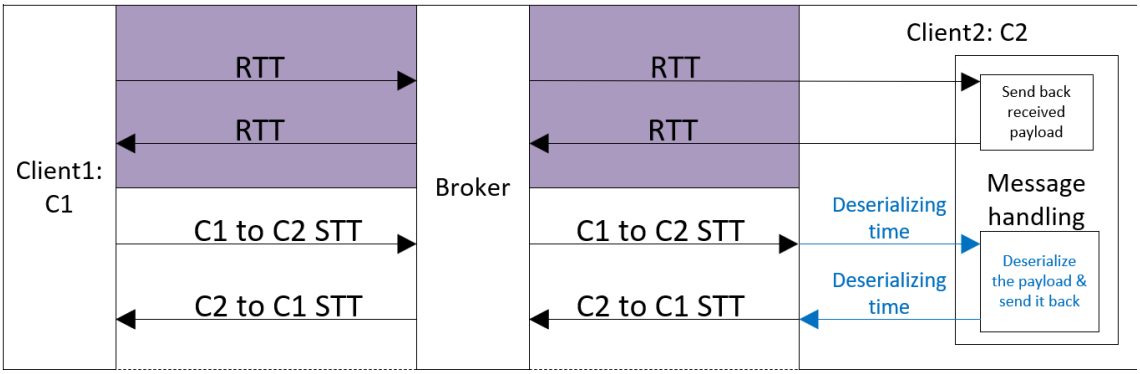


FIGURE 4. Broker with clients explaining times measured in more detail.

RTT is the time measured from the exact moment when Client1 starts sending the payload message to the assigned topic to when Client1 receives the message sent from Client2 after Client2 has received it from the topic that Client1 sent it to. This RTT time includes the duration for Client 2 to process and add the message payload sent from Client 1 into the returning message without deserialization.

As shown in FIGURE 4, the RTT measurement in the two-container setup excludes deserializing time but includes the time taken to send the payload back. In contrast, STT measurement in the one-container setup excludes the time for sending payload back, which is included in the deserializing time. The RTT mean or STT mean mentioned in this research will not include the time from deserializing if not mentioned otherwise.

4 RESULTS

4.1 Validating data

The data collected from two-container setup measurements reflects most accurately the conditions of the real platform. The next logical step is to ensure that STT times are not affected by additional steps, such as using memory-mapped files or differentiating processes within a single container. This is achieved by comparing the RTT results from two setups: the two-containers and the one-container setups. In FIGURE 5, the visualisation clearly illustrates the difference between the measurements for the two setups.

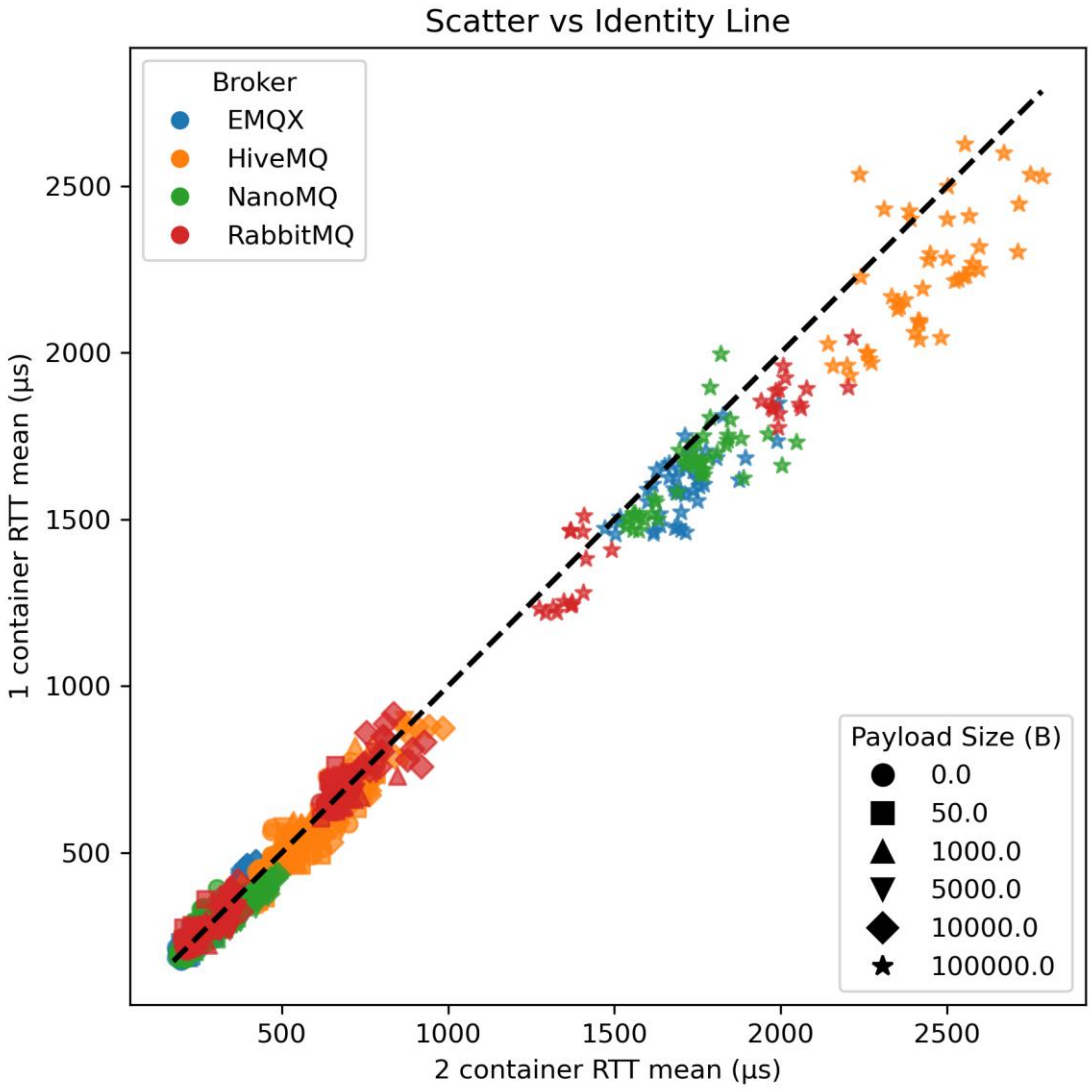


FIGURE 5. Scatter plot with identity line comparing one-container and two-container setups RTT means per broker and payload size.

It is evident that HiveMQ is having more peaks than the other brokers, and this difference becomes more pronounced with larger payload sizes, especially for HiveMQ. However, the scatter dots are generally close to the identity line, indicating that there is not a significant difference between the measurements. With larger payload size, the measurement points appear slightly more scattered, and the RTT for two-container setup tends to be marginally higher than for the one-container setup. This slight difference is due to the additional processing time required in the two-container setup to handle the payload in the MQTT message without deserializing it, from the moment Client2 receives the payload and sends back to Client1. In the one-container setup, this time is included with the payload deserializing time. The cap visualized in FIGURE 5 between measurements on RabbitMQ is from QoS 0 and QoS 1 and will be explained in more detail later in the results. To ensure that differences between RTTs are not too wide, a distribution of differences histogram is helpful as shown in FIGURE 6.

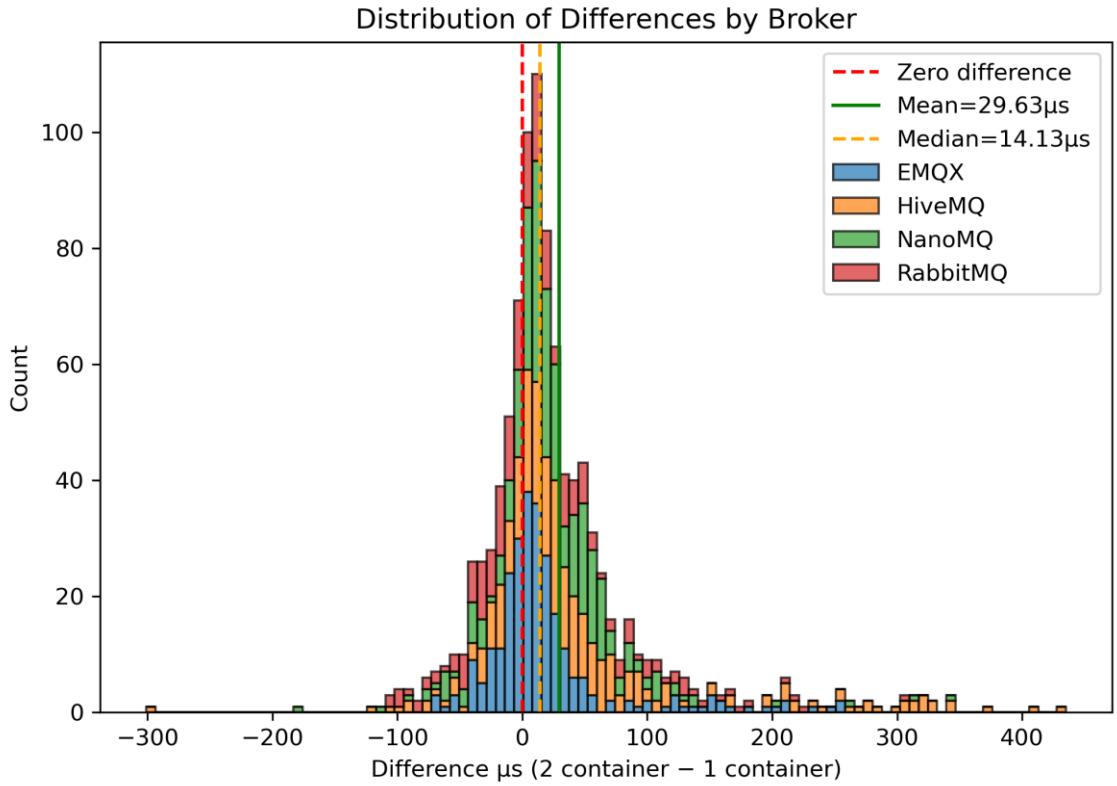


FIGURE 6. Distribution of differences by broker histogram.

For this type of measurements, it is expected to have differences between minimum and maximum values, with larger payloads resulting in wider the differences. This phenomenon is elaborated under the Conclusions section.

4.2 General results

The initial observation from analyzing the data of five identical measurement combination log files was that the first run after starting the container took longer than the subsequent runs. This phenomenon is illustrated in Appendix 2. This delay is expected due to the use of Just-In-Time (JIT) compiling in C#/.NET (Barrett etc. 2016, Microsoft 2024). JIT compiling occurs at runtime rather than before, which slows down the initial execution of the function (Barrett etc. 2016, Microsoft 2024). To evaluate whether the speed decrease is significant or acceptable, a more detailed examination of the timing data is necessary. The maximum difference between end times can be calculated by subtracting the minimum end time from the maximum end time of the measurement loop.

TABLE 1. Maximum difference between end times per broker and payload size using QoS 0 and binary payload format.

Broker	Payload size (B)	End difference (s)	Broker	Payload size (B)	End difference (s)
EMQX	0	0.60	HiveMQ	0	2.67
EMQX	50	0.55	HiveMQ	50	1.83
EMQX	1000	0.57	HiveMQ	1000	1.21
EMQX	5000	0.34	HiveMQ	5000	1.77
EMQX	10000	0.74	HiveMQ	10000	1.54
EMQX	100000	1.31	HiveMQ	100000	1.56
NanoMQ	0	0.46	RabbitMQ	0	0.86
NanoMQ	50	0.81	RabbitMQ	50	0.58
NanoMQ	1000	0.41	RabbitMQ	1000	0.42
NanoMQ	5000	0.29	RabbitMQ	5000	0.58
NanoMQ	10000	0.38	RabbitMQ	10000	0.55
NanoMQ	100000	0.42	RabbitMQ	100000	1.36

As observed in TABLE 1, the differences in end times range from 0.34s to 2.67s, which is generally tolerable. A delay of few seconds for the first ten thousand messages is not considered major, as it occurs only during the initial run (Barrett etc. 2016). HiveMQ appears to have most substantial and variable times, likely due to its use of the Java programming language, whereas other brokers are developed with more robust programming languages (F. Wang 2025, HiveMQ, n.d.).

4.3 RTT

The most critical requirement is to achieve communication with a RTT of 1 millisecond or less. An initial inspection of the results indicates that most RTT measurements for payload sizes under 100kB payload size are within the specified limit. A more detailed analysis, separation between brokers, payload sizes and QoS levels, is necessary to accurately determine the RTT performance, as illustrated in FIGURE 7.

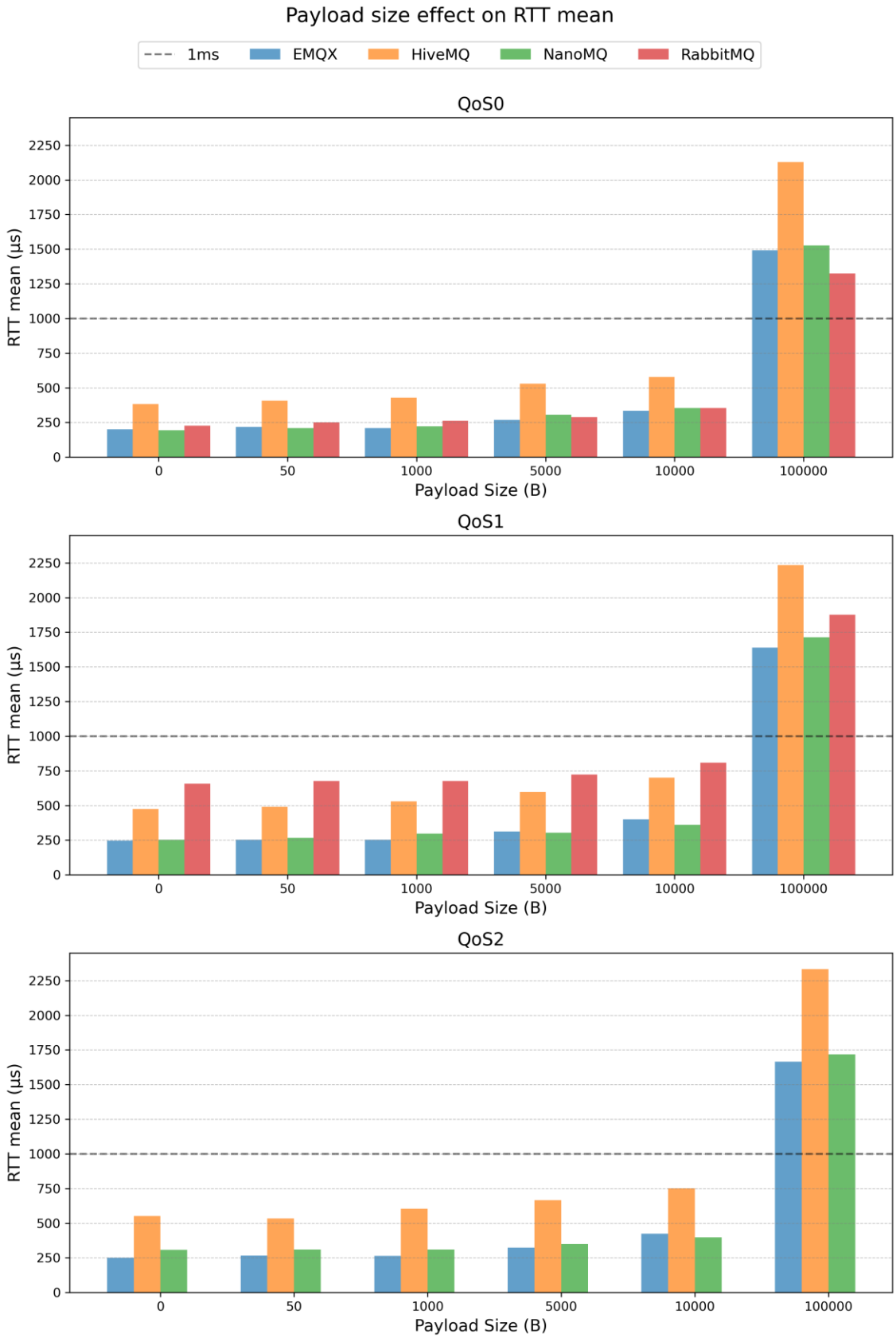


FIGURE 7. RTT means per brokers, payload sizes and QoS with 1ms line.

As shown in FIGURE 7, all brokers meet the company's requirement limit with all payload sizes but 100kB, with NanoMQ and EMQX achieving times closer to 0.5

milliseconds. FIGURE 7 shows HiveMQ and RabbitMQ having the most significant rise from QoS 0 to QoS 1 compared to other brokers.

As shown in the following figure, FIGURE 7, RabbitMQ and HiveMQ appear to have largest standard deviations, with HiveMQ displaying clearly more significant peaks. This is strongly related to the earlier explanation of HiveMQ having different underlying architecture than other brokers and instability of virtual machine resources like explained earlier in this document. (HiveMQ n.d., F. Wang 2025.) Comparing the results for EMQX and NanoMQ in FIGURE 8 reveals high degree of similarity, indicating that these brokers provide the most stable measurements under the given conditions.

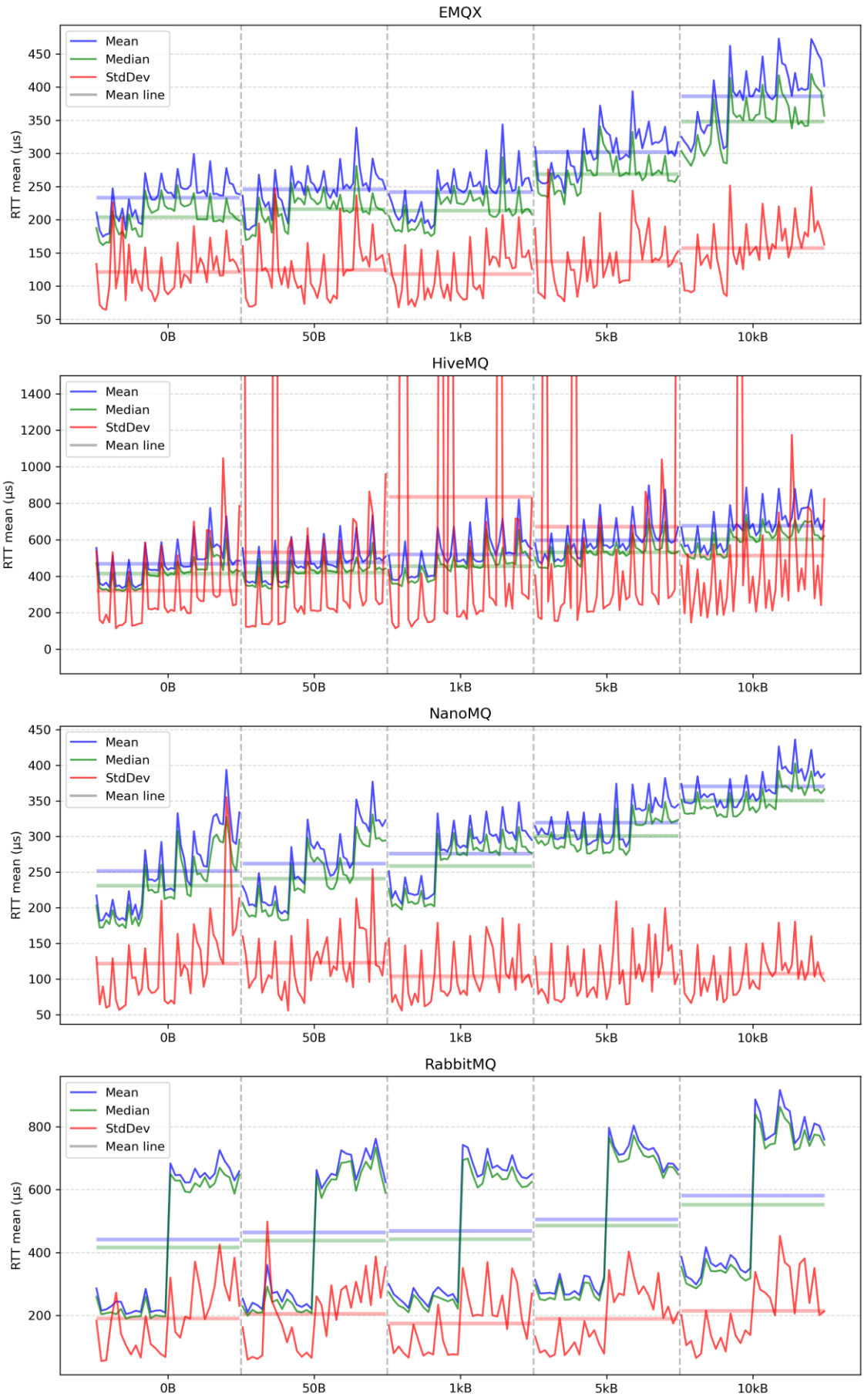


FIGURE 8. RTT mean, median and standard deviation measurement per broker and payload size.

4.4 QoS

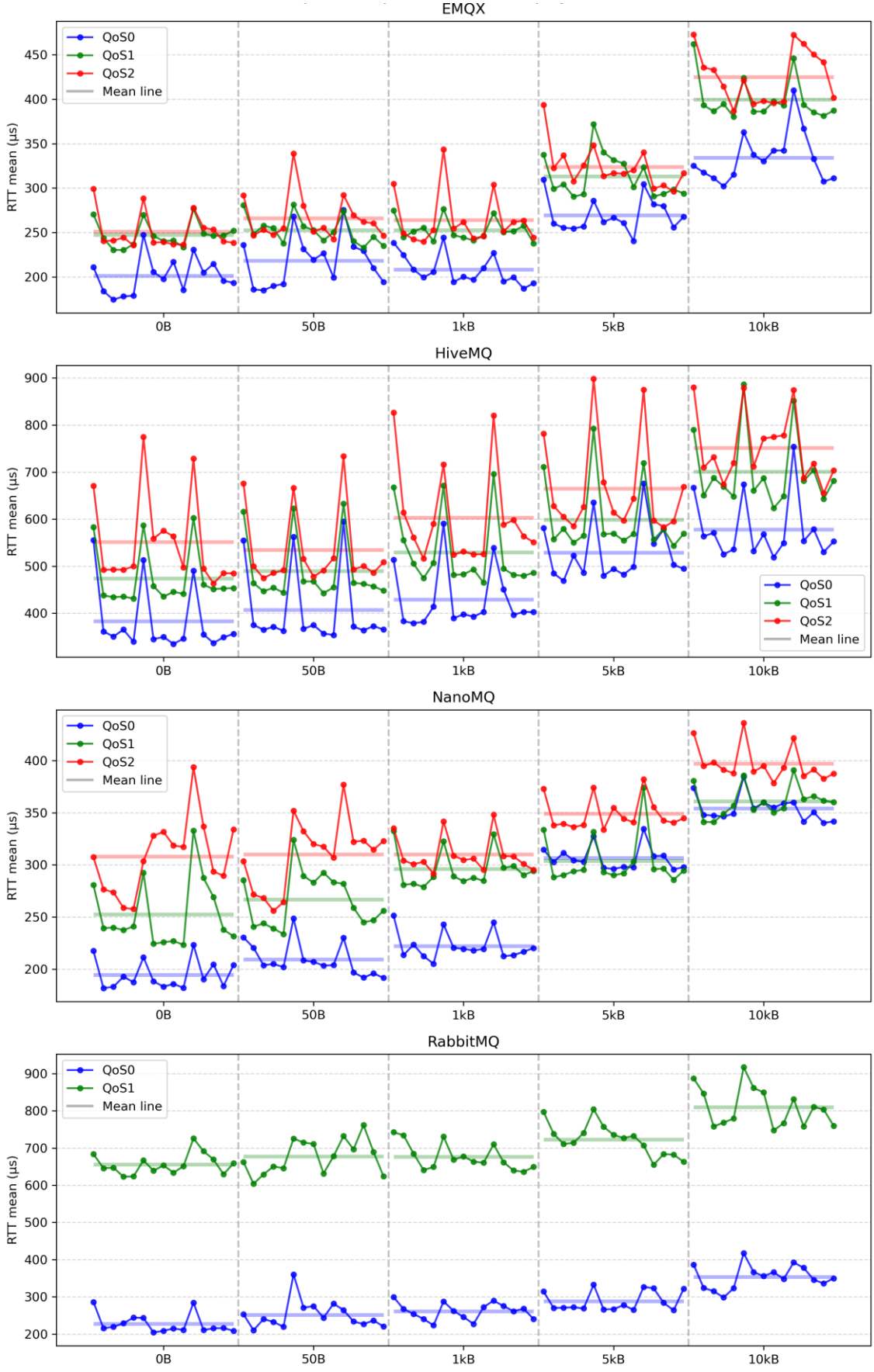


FIGURE 9. QoS RTT mean comparison per broker and payload size showing RTT mean for all formats in row.

As shown in FIGURE 9, RabbitMQ demonstrates the most significant difference between the measured QoS levels, with a difference of approximately 500 microseconds. In contrast, EMQX and NanoMQ show much smaller differences, around 100 microseconds. These differences are relatively insignificant, especially when considering the case where increasing the QoS level enhances reliability for message delivery and could eliminate the need for a validation message.

4.5 Payload size, format & deserializing time

Considering all the results presented thus far, it can be concluded that payload size effects RTT. Larger payload size result in higher RTT. This is illustrated in Appendices 3 to 6, which show separation between the sizes.

The deserializing measurements included in the dataset represent an interesting aspect that could benefit from further examination. The most straightforward approach is to calculate the payload format mean values from the actual log files and present them in a single figure with standard deviation whisker to see how big the level is.

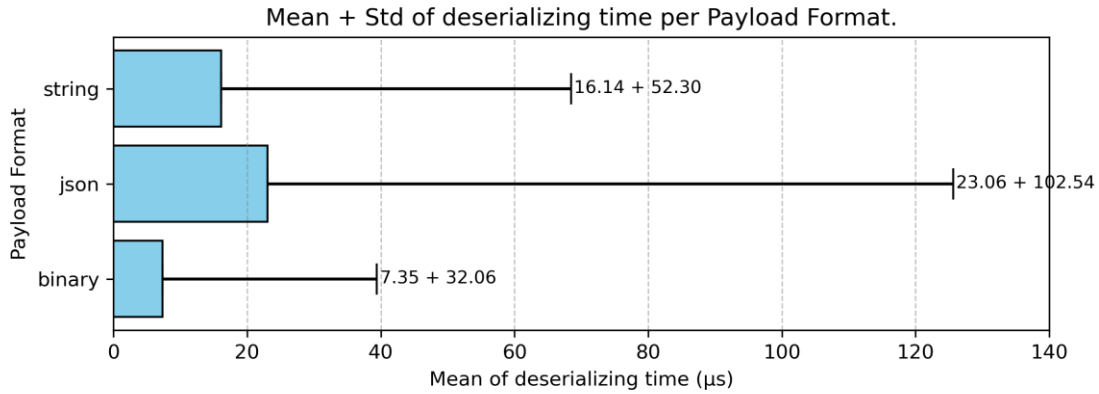


FIGURE 10. Mean and standard deviation of deserializing time per payload format.

FIGURE 10 illustrate that the time used for deserializing the message payload is minimal. It also shows a clear difference between binary, JSON and string for-

mats, as expected (R. Maltsev & R. Amin 2024). However, the analysis of Appendices suggest that the deserializing time is so small that it is overshadowed by the variability in the RTT mean.

Appendices 3 to 6 demonstrate that the impact of payload format on RTT is minimal when deserializing time is included. The primary effect of payload format is observed in the deserializing time, rather than in the overall RTT. This indicates that while different payload formats may influence how quickly data can be processed, their influence on RTT, which includes deserializing time, is relatively negligible.

4.6 STT

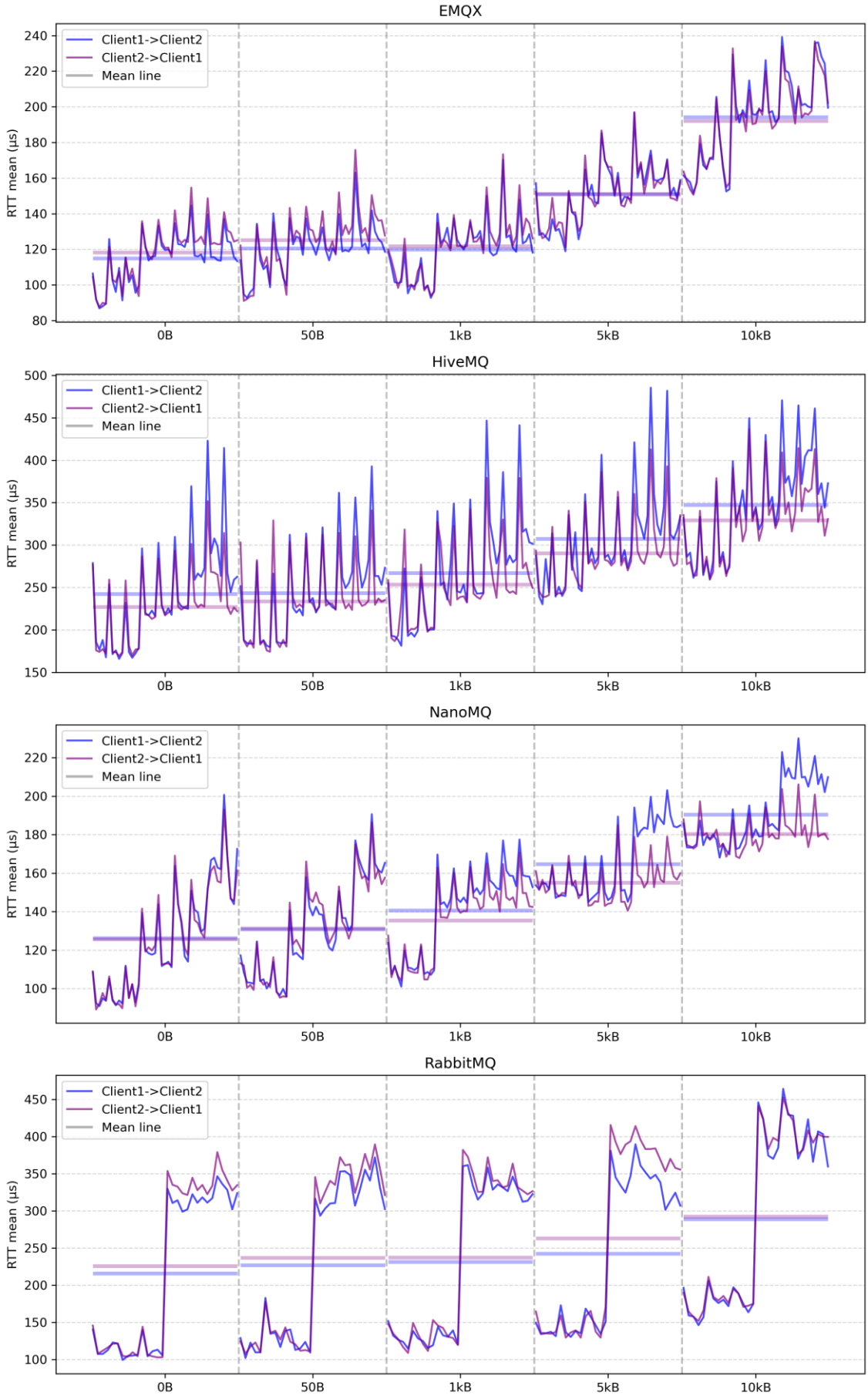


FIGURE 11. Difference between Client1 to Client 2 and Client2 to Client1 STT.

The difference between the STTs in FIGURE 11 appears to be negligible, which is the expected result (Almes etc 1999). The STTs are easily comparable, as the mean lines almost overlap, considering the peaky nature of these measurements. When comparing the brokers in FIGURE 11, note that the y-axis values vary, making the negligible differences easier to observe.

5 CONCLUSION

The results and measurements presented are consistent with existing literature and industry standards. The effects of QoS and payload size on RTT means are clearly demonstrated, while the impact of payload format is limited to deserializing time.

The results show that optimizing payload size is crucial for minimizing RTT. Using 100kB payloads results in considerably longer RTT times, whereas the differences between smaller payload sizes are negligible. Although there are differences in deserializing times across payload formats, these differences are minimal considering the peaky nature of RTT means and microsecond-level measurements. The natural use case for MQTT involves sending messages with payloads that trigger actions, such as refreshing data on a UI, logging, or setting instrument outputs, which allows for the use of small message payload sizes.

As expected, QoS levels affect RTT, and the results confirm this relationship. FIGURE 9 shows that the measured differences are low for most brokers, except for RabbitMQ, which demonstrates a significant difference between QoS 0 and QoS 1 (EMQX Team (3) 2023). This phenomenon arises because RabbitMQ uses MQTT through a plugin, translating MQTT protocol messaging from AMQP (RabbitMQ n.d.). This also explains why RabbitMQ does not have QoS 2, as AMQP does not support the exact “exactly once” semantics (RabbitMQ n.d.). In contrast, the other brokers used in this research are MQTT-native, processing straightforward PUB/PUBACK paths directly (HiveMQ n.d., Wang 2025).

The initial delay observed when running a function after starting a container is noteworthy but not restrictive. A couple seconds delay on the first function call is insignificant for day-to-day usage but should be considered when evaluating golden sample performance. (Barrett etc. 2016).

Based on this research, the most optimized broker and message configuration combination involves using either NanoMQ or EMQX brokers. The difference in performance between the QoS levels is minimal, making it realistic to upgrade

the level from 0 to 2. However, it is not conclusive whether either of these brokers can handle all test platform communication. Therefore, further investigation of broker stress test with multiple clients evident. The MQTT Explorer software was found to affect RTT by burdening the broker, creating a question of how multiple clients or client subscribing to multiple topic effect on RTT. Future research could also examine the effect of loop amounts, potential impact of another broker and broker configurations optimizations.

TABLE 2. Results of comparing mean of RTT means to mean of RTT medians in microseconds.

Mean of Means	Mean of Medians	Mean of mean – mean of median result	Result % of mean of Means
628.31	572.66	55.65	8.86

After evaluating if the results are actually correct and usable a notable finding appeared. As shown in TABLE 2, the mean of RTT means is about 9 percent higher than the mean of RTT medians. This indicates that the mean is likely influenced by the high peaks. The median, which does not add as much weight on the peaks as the mean, provides a more robust value for evaluating the central estimate. It is important to acknowledge this when comparing the results, and considering the median as an alternative measure could be a better way of presenting data in this document's results. An example of median as the visualized value instead of mean in FIGURE 6 is added as Appendix 7. Another observation from this research is that some of the figures are on high level and generalizing and could provide more insight and possible alternative perspectives highlighting trends that are not visible in these figures. Using more variable set of KPI values would add different point of views and deeper and informative results.

The research involved two containerized clients that communicated with each other's through a containerized broker. RTT measurements were implemented to simulate message validation. The MQTT communication protocol achieved RTT of less than 1 millisecond across all brokers and QoS levels with payload sizes

ranging from 0 to 10kB. Each of these factors contributed to the research meeting the company's requirements.

At the start of this research, the plan was to research other communication protocols if MQTT was not meeting the requirements. Since MQTT has proven sufficient, the research question about alternative communication protocols performance remains open. Simple examples of other communication protocols are gRPC and ZeroMQ. Further investigation into optimizing the hardware setups, alternatives for virtual machine, and understanding the impact of networking over specific firewalls on RTT could also add value.

This research addressed the company’s requirements while leaving open several new research question possibilities, maintaining the scope of the research on appropriate level. The research questions could have been integrated, but this would have cost by the research scope becoming too broad. At the bachelor level, the research scope was appropriate, and the company’s requirements were answered. Future research on designing a refined architecture can continue as the company sees fit.

SOURCES

Almes, G., Kalidindi, S., Zekauskas, M. 1999. A One-way Delay Metric for IPPM. Website. Accessed 2.11.2025. https://data-tracker.ietf.org/doc/html/rfc2679

Ansari, D. 2023. Serving Millions of Clients with Native MQTT. Website. Accessed 2.11.2025. https://www.rabbitmq.com/blog/2023/03/21/native-mqtt

Barrett, E., Bolz, C. F., Killick, R., Knight, V., Mount, S. & Tratt, L. 2016. PDF-file. Accessed 3.10.2025. https://theunixzoo.co.uk/pubs/pdf/barrett2016warmup.pdf

Docker. n.d. Use containers to Build, Share and Run your applications. Website. Accessed 3.10.2025. https://www.docker.com/resources/what-container/

Docker Hub. 2025. EMQX overview. Website. Accessed 3.10.2025. https://hub.docker.com/r/emqx/emqx

EMQX docs. n.d. Design for EMQX Clustering. Website. Accessed 3.10.2025. https://docs.emqx.com/en/emqx/latest/design/clustering.html

EMQX Team (1). 2023. MQTT Maximum Packet Size Explained and Example | MQTT 5 Features. Website. Accessed 3.10.2025. https://www.emqx.com/en/blog/best-practices-of-maximum-packet-size-in-mqtt

EMQX Team (2). 2023. A Beginner's Guide to MQTT Performance Testing. Website. Accessed 2.11.2025. https://www.emqx.com/en/blog/a-beginner-guide-to-mqtt-performance-testing

EMQX Team (3). 2023. MQTT Performance Benchmark Testing: EMQX Single Node Message Latency & Response Time. Website. Accessed 2.11.2025. https://www.emqx.com/en/blog/mqtt-performance-benchmark-testing-emqx-single-node-message-latency-response-time

EMQX Team. 2024. How to process JSON, Hex, and Binary Data in MQTT. Website. Accessed 3.10.2025. https://www.emqx.com/en/blog/how-to-process-json-hex-and-binary-data-in-mqtt

EMQX Team. 2025. Mastering MQTT: The Ultimate Beginner's Guide for 2025. Website. Accessed 3.10.2025. https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt

HiveMQ, n.d. Highlighted Open Source Products. Website. Accessed 3.10.2025. https://www.hivemq.com/community/open-source/

Jin, M. 2025. Open MQTT Benchmarking Comparison: MQTT Brokers in 2023. Website. Accessed 2.11.2025. https://www.emqx.com/en/blog/open-mqtt-benchmarking-comparison-mqtt-brokers-in-2023

Maltzev, E. & Amin, R., 2024. Impact of Serialization Format on Inter-Service Latency. PDF-file. Accessed 2.11.2025. https://www.researchgate.net/publication/387163684_Impact_of_Serialization_Format_on_Inter-Service_Latency

Microsoft. 2024. Compile CIL to native code. Website. Accessed 3.10.2025. https://learn.microsoft.com/en-us/dotnet/standard/managed-execution-process#compile-cil-to-native-code

RabbitMQ. n.d. MQTT Limitations. Website. Accessed 3.10.2025. https://www.rabbitmq.com/docs/mqtt#limitations

RabbitMQ. n.d. Plugins. Website. Accessed 3.10.2025. https://www.rabbitmq.com/docs/plugins

Wang, F. 2025. Comparison of Open Source MQTT Brokers 2025. Website. Accessed 3.10.2025. https://www.emqx.com/en/blog/a-comprehensive-comparison-of-open-source-mqtt-brokers-in-2023

Zhou, Z. 2025. MQTT QoS 0, 1, 2 Explained: A Quickstart Guide. Website. Accessed 3.10.2025. https://www.emqx.com/en/blog/introduction-to-mqtt-qos

APPENDICES

Appendix 1. 2-container-results.csv EMQX-data

Broker	Payload size	QoS	Payload format	Mean	Median	StdDev	Min	Max	Start µs	End µs
EMQX	0	0	binary	248.508	215.4	243.942	122.1	12221	11332.3	2633752.4
EMQX	0	0	binary	197.853	179.55	167.164	101.9	14347.3	926.9	2068938.5
EMQX	0	0	binary	198.524	180.5	80.713	116.3	2643.7	693.3	2081644.4
EMQX	0	0	binary	200.753	181.6	235.714	101.5	17835.6	1068	2086124.5
EMQX	0	0	binary	195.719	178	209.444	112.4	14323	1209.3	2033828.1
EMQX	0	0	json	223.666	200.5	233.618	122.7	19082.2	10488.9	2359721.3
EMQX	0	0	json	183.843	168.4	75.382	105.4	2216.3	773	1893965.8
EMQX	0	0	json	190.663	171.8	73.522	121.9	1913	785.9	1961342.5
EMQX	0	0	json	181.749	164.6	78.593	116.7	3133.9	1388.8	1875770.3
EMQX	0	0	json	181.725	168.1	64.999	126.8	2169.1	852.7	1871699.1
EMQX	0	0	string	225.857	193.4	274.627	130.7	15207.4	11362.1	2376688.3
EMQX	0	0	string	192.347	176.4	79.53	119.2	2506.7	742.7	1982659.4
EMQX	0	0	string	190.922	173.7	79.737	123	2925.6	988.5	1968869.9
EMQX	0	0	string	187.604	170.6	86.972	120.2	3326.4	840.4	1931209.5
EMQX	0	0	string	195.7	177.9	73.596	123.2	2429.1	762.5	2016766.9
EMQX	0	1	binary	268.621	242.8	155.988	153.3	11816.7	11855.5	2811796.2
EMQX	0	1	binary	251.483	227.1	97.557	133.6	3462.9	727	2585762.2
EMQX	0	1	binary	246.401	226.1	77.242	146.2	2408.6	862.8	2531617.8
EMQX	0	1	binary	249.708	226.8	94.482	143.6	4259	748.5	2565446.3
EMQX	0	1	binary	237.92	220.7	106.388	132.1	7287.3	650.6	2445457.4
EMQX	0	1	json	270.644	243.4	165.984	152.4	12757.7	12809.4	2835678.8
EMQX	0	1	json	248.379	224.6	96.775	142.4	3604.2	723.7	2552767.2
EMQX	0	1	json	246.588	225.5	103.67	143.2	3120.9	660.9	2535150.2
EMQX	0	1	json	240.66	219.7	94.795	142.3	3167.9	899.1	2475336.7
EMQX	0	1	json	235.119	219.1	71.054	147.7	1942	725.6	2420957.8
EMQX	0	1	string	277.136	253	164.44	123.2	11734.4	11772.4	2903076.7
EMQX	0	1	string	242.94	223	90.887	130.9	2874.7	576.2	2501229.3
EMQX	0	1	string	246.64	228.1	89.212	147	2807.9	871.4	2537157.5
EMQX	0	1	string	238.823	221.7	77.887	130.5	2325.7	713	2457920.9
EMQX	0	1	string	243.394	219.1	93.979	132.7	2357.5	500.8	2505108.1
EMQX	0	2	binary	325.86	265.05	260.207	141.1	11473.8	11514.5	3423377.2
EMQX	0	2	binary	274.009	226.4	161.506	136.6	4111.3	687.6	2827894.1
EMQX	0	2	binary	269.002	222.5	162.188	137.7	5620.7	740.5	2770653.1
EMQX	0	2	binary	267.898	221.1	169.274	136.7	5313.8	827.8	2761614.4
EMQX	0	2	binary	260.812	220.3	145.082	129.6	4872.3	1520.8	2684247.9
EMQX	0	2	json	298.622	244.3	195.007	126.5	10966.4	11004.2	3155601

EMQX	0	2	json	253.867	209.7	131.225	125.8	2627.2	1031.8	2616300.9
EMQX	0	2	json	245.157	206.5	113.465	119.5	2371.4	881	2527216.1
EMQX	0	2	json	258.327	211.2	146.618	134.4	4573.8	620.5	2661169
EMQX	0	2	json	246.454	206.7	128.846	125.5	3373.2	644.8	2544273.3
EMQX	0	2	string	324.779	262.75	228.991	137	9820.6	9859.2	3433729
EMQX	0	2	string	278.494	233.2	160.355	132.8	5708.1	786.8	2869242
EMQX	0	2	string	270.96	227.45	235.039	123.4	16391	938.5	2793642.1
EMQX	0	2	string	295.663	238.7	282.11	137.3	14165	1774.4	3044596.3
EMQX	0	2	string	263.9	217.2	147.14	124.5	3240.3	776.2	2722832.8
EMQX	50	0	binary	249.806	217	271.435	117.4	14230.4	11309.4	2616101.5
EMQX	50	0	binary	213.949	190.4	127.153	129.9	6980.6	850.3	2203901.4
EMQX	50	0	binary	200.591	183.1	85.344	124.8	2420.9	736.8	2065505.9
EMQX	50	0	binary	227.587	194.3	103.259	131.1	3484.2	728.3	2346890.8
EMQX	50	0	binary	210.42	192.2	76.592	120.5	2168.3	674.5	2161207.9
EMQX	50	0	json	225.702	203	260.57	134.8	24612.6	24664.2	2377602.2
EMQX	50	0	json	196.303	184.3	73.693	142.5	3488.5	625.8	2021714.3
EMQX	50	0	json	198.577	182.5	68.014	135.8	1840.3	618.3	2040279.4
EMQX	50	0	json	203.12	186.1	70.404	120.5	1823.7	762.1	2085858.9
EMQX	50	0	json	192.892	183.6	53.472	128.4	1451	727.9	1983157.3
EMQX	50	0	string	237.954	209	152.067	123.4	10412	10452.4	2502355.9
EMQX	50	0	string	209.121	185.5	103.846	133.2	3624.6	725.1	2151430.1
EMQX	50	0	string	212.273	184.5	109.498	114.1	4120.7	680.5	2185115.4
EMQX	50	0	string	201.216	183.2	78.534	120.5	2061.6	746.4	2071370.3
EMQX	50	0	string	203.505	184.5	84.863	131.6	2968.1	780.2	2095717.7
EMQX	50	1	binary	306.637	275.2	225.833	165.3	11488.9	10989.7	3204487.7
EMQX	50	1	binary	268.991	247.1	195.839	148.1	13175.3	816.6	2763339.8
EMQX	50	1	binary	265.796	240.4	118.871	146	3824.5	721.7	2735311.9
EMQX	50	1	binary	267.716	240.3	116.137	151.1	3876.5	805.5	2752500.4
EMQX	50	1	binary	258.355	234.2	107.485	155.2	4140.3	1141.6	2658078.5
EMQX	50	1	json	284.793	256.5	277.103	161	25549.3	25588	2984380
EMQX	50	1	json	258.195	235.6	117.201	138	3993	596	2650649.9
EMQX	50	1	json	265.162	234.5	138.265	137.1	3816.4	712.9	2721456.7
EMQX	50	1	json	261.338	242.7	90.801	164.6	2814.8	638.2	2685256.8
EMQX	50	1	json	252.722	233.2	102.897	145.4	3240.2	777.9	2600138.6
EMQX	50	1	string	284.586	259	151.26	154.5	10057.6	10101.8	2991395.6
EMQX	50	1	string	248.828	229.2	86.159	137.8	2516.2	966	2558496.2
EMQX	50	1	string	253.526	236.3	82.756	152.6	2384.1	799	2604291.7
EMQX	50	1	string	249.424	231.1	85.186	150.9	2490.4	692.1	2561738.5
EMQX	50	1	string	241.967	225.1	81.092	143.4	2610.7	805.5	2489547.8
EMQX	50	2	binary	326.446	269.4	247.969	145	13922.3	13978.5	3422721
EMQX	50	2	binary	294.97	249.8	170.474	156.1	9272.5	1057.1	3036835.1
EMQX	50	2	binary	289.531	240.9	228.186	145.4	12083.6	665.6	2982602
EMQX	50	2	binary	295.438	246.1	147.293	147.4	3681.3	1042.6	3038887.6
EMQX	50	2	binary	291.816	241	221.673	140.3	13966.8	2506.2	3003687.4
EMQX	50	2	json	308.678	245.4	356.559	141.1	26398.1	26432.1	3242706.2
EMQX	50	2	json	270.08	223.4	136.845	136.7	2389.1	649.1	2776117.8

EMQX	50	2	json	259.358	217.9	155.198	123.6	8418	988.6	2670438.7
EMQX	50	2	json	252.093	214.35	116.12	137.6	2232	630.7	2597518.5
EMQX	50	2	json	245.993	211.2	144.393	140.2	7398.7	844.9	2532605.1
EMQX	50	2	string	299.308	243.6	197.836	144.9	11824.8	11862.1	3152750.3
EMQX	50	2	string	251.95	210.8	125.368	137.1	2950.1	601.6	2592185.6
EMQX	50	2	string	267.658	225.65	127.533	137.2	2579.2	659.5	2752658.5
EMQX	50	2	string	252.769	213.3	125.761	134.5	3120.4	719.7	2599310.3
EMQX	50	2	string	256.298	214.7	122.737	132.5	2126.5	1056.9	2635802.1
EMQX	1000	0	binary	244.602	224.6	147.839	135.3	11086.8	11124.3	2598606
EMQX	1000	0	binary	195.394	179.1	76.015	131.6	2200.5	1018.2	2029851.5
EMQX	1000	0	binary	200.603	185	69.363	133.4	1914.3	718.9	2083324.2
EMQX	1000	0	binary	202.702	182.8	89.181	131.5	2873.7	604.3	2102812.7
EMQX	1000	0	binary	199.006	184.7	59.953	130.7	2129.5	626	2067165.8
EMQX	1000	0	json	248.997	217.8	288.636	134.8	25339.3	25388.5	2632190.4
EMQX	1000	0	json	211.972	191.5	85.329	130.2	3011.7	838.8	2194658.1
EMQX	1000	0	json	209.954	189.2	97.724	141.5	3100.9	770.5	2176571.6
EMQX	1000	0	json	210.599	188.8	118.746	128.6	4230.3	652.3	2182234.5
EMQX	1000	0	json	201.482	185.4	83.559	135	2575.2	767.4	2088982.8
EMQX	1000	0	string	252.352	218.3	156.75	135.9	10179.4	10234.9	2666097.6
EMQX	1000	0	string	214.333	187.5	103.948	114	2916	588.9	2223790.3
EMQX	1000	0	string	211.094	184.5	97.295	126	2776.4	559.6	2185441.3
EMQX	1000	0	string	219.481	192.3	106.842	131.2	2139.2	623.7	2271644
EMQX	1000	0	string	200.853	182.5	81.407	125.6	3777.1	910.2	2081061.1
EMQX	1000	1	binary	272.745	245.8	156.939	154.7	10242.6	10280.8	2873395.9
EMQX	1000	1	binary	240.43	222.4	94.65	147.6	2834	542.4	2490505
EMQX	1000	1	binary	249.994	228.4	85.466	149.5	2092.4	607.2	2585092.4
EMQX	1000	1	binary	248.545	225.9	96.665	154.3	3682.5	761.3	2574105.7
EMQX	1000	1	binary	245.498	226.1	87.294	136.2	2570.4	791.9	2544436.3
EMQX	1000	1	json	289.1	259.2	271.348	163.4	23422.9	23458.4	3043155.9
EMQX	1000	1	json	248.729	228.15	117.477	156.9	4981	1083.1	2575752.8
EMQX	1000	1	json	258.973	230.3	136.224	153.5	7451.2	707.2	2678169.1
EMQX	1000	1	json	249.976	227.7	93.182	154.2	2281.9	681.7	2589798.3
EMQX	1000	1	json	255.417	234	106.377	159.3	3493.1	561.5	2640606.5
EMQX	1000	1	string	287.339	255.1	183.35	166.3	10614.2	10650.9	3025588.4
EMQX	1000	1	string	254.414	231.2	96.08	152.3	2391.2	1299.1	2634776.9
EMQX	1000	1	string	252.007	229.9	96.758	148	3561.5	1423.1	2610564.4
EMQX	1000	1	string	252.415	232.8	106.575	151.3	4275.5	886.9	2612113.3
EMQX	1000	1	string	243.797	225.6	101.527	157.4	3523.2	673.3	2524399.3
EMQX	1000	2	binary	286.191	238.9	197.096	137.2	10488.4	10528.4	3030725.9
EMQX	1000	2	binary	260.87	214.4	154.887	136.5	7927.7	2556.8	2701357.9
EMQX	1000	2	binary	248.268	210.8	119.854	138	3607.5	668.1	2575332.5
EMQX	1000	2	binary	253.37	211.5	145.817	138.4	5196.8	750.6	2628112.9
EMQX	1000	2	binary	249.113	209.9	157.158	136.3	5272.3	622.8	2584308.5
EMQX	1000	2	json	304.401	249.4	285.176	140.6	22992.6	23031.1	3213240.3
EMQX	1000	2	json	265.242	219.5	149.694	139.8	3603.6	1325.5	2753694.3
EMQX	1000	2	json	251.917	214.25	124.229	134.9	3237.7	798.4	2613545.2

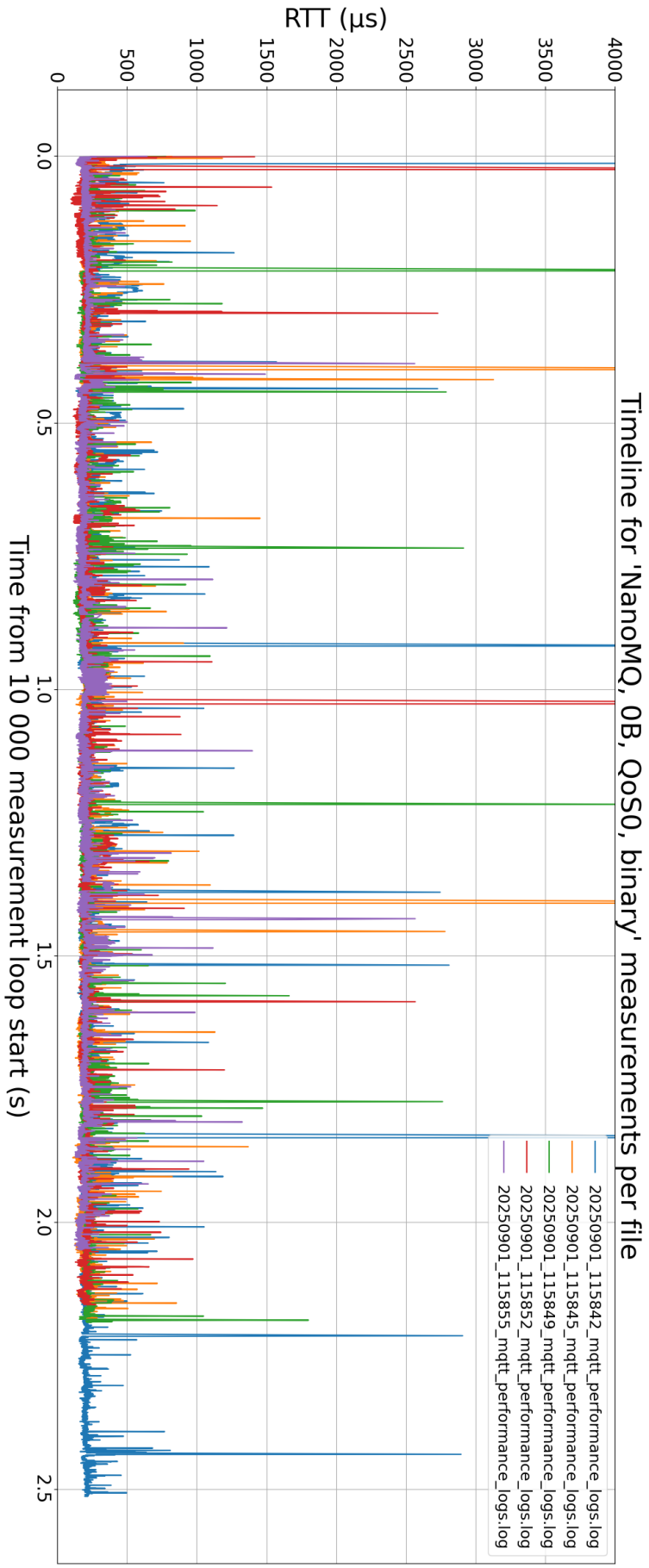
EMQX	1000	2	json	242.289	208.45	133.193	137.5	4290.6	794.7	2513485.5
EMQX	1000	2	json	247.1	212	118.381	134.9	2794.8	736.4	2563742.5
EMQX	1000	2	string	301.945	248.5	187.233	135.6	10357.9	10395.6	3191057.9
EMQX	1000	2	string	271.653	225.3	150.878	141.6	7177.9	690.9	2816208.4
EMQX	1000	2	string	262.365	220.3	133.8	137.3	2743.3	1590.1	2724747.8
EMQX	1000	2	string	253.117	217.8	118.127	136.4	2688	750.9	2626697.2
EMQX	1000	2	string	255.459	214	134.811	139.7	3974.5	643	2649430
EMQX	5000	0	binary	288.881	263.85	158.576	163.8	10496.7	10532.8	3036498.2
EMQX	5000	0	binary	282.2	267.25	106.176	158.7	3011.4	815.3	2925577.8
EMQX	5000	0	binary	277.583	252.3	120.753	164.9	3560.3	2211.5	2874722.9
EMQX	5000	0	binary	271.659	246.15	118.706	156	3664.2	2519.9	2814055.9
EMQX	5000	0	binary	259.439	236.8	109.731	148.9	3647.5	803	2691831.1
EMQX	5000	0	json	303.489	275.2	281.593	183.1	25151.4	25194.7	3178360.6
EMQX	5000	0	json	282.239	263.4	97.299	155.3	2264.4	719.8	2923352.8
EMQX	5000	0	json	283.591	267.4	92.565	168.4	2766.7	936.2	2937947.3
EMQX	5000	0	json	294.346	269.3	122.793	168	2794.4	897.8	3049327.4
EMQX	5000	0	json	274.599	254.75	111.841	167.7	3646.1	829.6	2846318.8
EMQX	5000	0	string	285.309	268.2	157.49	177.5	10511.9	10548.6	2994971.7
EMQX	5000	0	string	267.802	249.2	102.058	165.1	4372.2	826.5	2777926
EMQX	5000	0	string	270.82	248.8	104.62	166.7	3865.8	862	2805514.8
EMQX	5000	0	string	294.625	272.1	198.852	166.7	10429.8	2246	3060780.2
EMQX	5000	0	string	276.064	254.5	116.973	160.1	3282.3	1103.7	2864257.2
EMQX	5000	1	binary	325.866	293.6	199.973	187.4	12239.3	12278.1	3406263
EMQX	5000	1	binary	294.665	268.6	122.317	184.6	3335.5	690.5	3035599.3
EMQX	5000	1	binary	310.454	284.7	117.229	198.8	2901.3	812.9	3198671.1
EMQX	5000	1	binary	291.897	271	88.227	173.2	2220.2	804.7	3007419.6
EMQX	5000	1	binary	292.843	270.5	128.453	181.9	8451.2	737.8	3017676
EMQX	5000	1	json	357.054	320.6	284.569	195.8	24138	24173.6	3719998.3
EMQX	5000	1	json	339.958	303.7	148.714	182	3459.1	829.2	3499500.3
EMQX	5000	1	json	327.943	292.2	151.593	187.9	2839.8	725.8	3375871.2
EMQX	5000	1	json	328.558	291.65	149.136	195.4	4251.1	1122.4	3379714.3
EMQX	5000	1	json	326.917	287.5	162.975	188.2	3759.3	695.4	3368975
EMQX	5000	1	string	393.262	351	287.601	176.5	13806.3	12903.2	4089040.5
EMQX	5000	1	string	338.575	310.45	154.666	201.2	7158.1	999.2	3486258.7
EMQX	5000	1	string	316.456	292.85	121.751	189.1	3952.7	1530.8	3265352.6
EMQX	5000	1	string	310.242	280.1	143.194	174.6	4681.7	4682.6	3195794.1
EMQX	5000	1	string	307.795	275.85	224.604	192.2	15061	1296.4	3170342.4
EMQX	5000	2	binary	405.601	345.4	320.085	188.5	13802.9	12081.6	4253285.4
EMQX	5000	2	binary	356.331	310.4	179.201	189.3	9149.1	2656.6	3689726.8
EMQX	5000	2	binary	313.387	273.15	167.902	173.1	6225.8	1199	3234498.5
EMQX	5000	2	binary	331.106	278.9	182.173	179	5398.4	631.8	3413480.6
EMQX	5000	2	binary	320.336	274.5	157.539	181.6	4520.7	880.7	3308454.8
EMQX	5000	2	json	360.807	306.8	292.414	170.8	23784.7	23832.3	3779215.4
EMQX	5000	2	json	320.31	274.1	146.423	177.8	4404.7	892.9	3309926.5
EMQX	5000	2	json	320.373	278.4	146.107	183.5	4482.8	861.1	3311723.5
EMQX	5000	2	json	345.067	280.55	249.155	175.8	8049.5	1880.3	3557151.1

EMQX	5000	2	json	316.353	276.2	154.9	176.3	5713.9	1459.9	3269238.9
EMQX	5000	2	string	357.848	304.2	214.254	172.3	12651.9	12689.1	3737774.9
EMQX	5000	2	string	331.517	279.6	152.766	186	3843.3	1168.9	3424196.1
EMQX	5000	2	string	339.352	284.7	166.322	178.7	4620.2	787	3505682.1
EMQX	5000	2	string	319.897	271.15	171.067	172.3	5539.5	2780.5	3303417.4
EMQX	5000	2	string	327.633	278.8	140.012	187.4	2420.7	743.7	3386234.8
EMQX	10000	0	binary	372.814	336.8	181.656	212	10651.7	10729.4	3861778.8
EMQX	10000	0	binary	321.583	293.8	100.413	215.5	2740.5	735.8	3299983.2
EMQX	10000	0	binary	304.011	285.7	66.049	207.7	1577.8	650.2	3119358.8
EMQX	10000	0	binary	327.169	303.85	89.78	228.3	2901.4	823.7	3356017.9
EMQX	10000	0	binary	305.291	281.8	109.508	223.5	6557.9	702.5	3127998.9
EMQX	10000	0	json	367.523	329.7	275.79	205.1	23815.9	23846.4	3790882
EMQX	10000	0	json	338.099	302.8	124.844	210.9	2737.1	928.3	3463536.4
EMQX	10000	0	json	345.37	308.1	137.821	211.8	4563.8	729.7	3536474.7
EMQX	10000	0	json	340.738	308.8	131.482	215.2	2972.7	730.6	3487136.5
EMQX	10000	0	json	334.095	301.7	125.457	222.6	3117.7	741.1	3421320.4
EMQX	10000	0	string	367.108	332.35	255.745	209.9	16712.5	9982.5	3793608.1
EMQX	10000	0	string	329.332	295.2	116.196	215.4	2382.4	705.8	3376936.6
EMQX	10000	0	string	334.737	302.3	122.471	207.3	3715.1	671.1	3424936.3
EMQX	10000	0	string	321.402	292.8	99.829	212.8	3458.5	1030.2	3300206.1
EMQX	10000	0	string	330.553	297.9	123.768	219.7	5299.5	758.4	3385200.1
EMQX	10000	1	binary	394.401	358.55	171.753	238.5	10230.5	10266.4	4067774.9
EMQX	10000	1	binary	378.952	349.1	125.189	229.2	2978.3	850.3	3863367.5
EMQX	10000	1	binary	374.148	341.2	116.326	222.7	2399.9	664.4	3817932.3
EMQX	10000	1	binary	391.541	366.4	113.701	232.9	3082.2	796.4	3990712.9
EMQX	10000	1	binary	364.106	336.15	111.916	227	2916.3	913.4	3714732.8
EMQX	10000	1	json	403.579	364.3	312.235	223	25530.5	25565.6	4152231.3
EMQX	10000	1	json	359.061	335.9	95.672	237.6	2222.8	772.4	3664420.6
EMQX	10000	1	json	366.724	337.3	104.749	232.5	2023.8	796.9	3741195.6
EMQX	10000	1	json	364.049	338.7	96.604	237.4	1710.5	1375.6	3712720.2
EMQX	10000	1	json	365.669	337	121.34	234.8	4927.6	866.5	3728808
EMQX	10000	1	string	389.537	361.7	147.383	243.3	9790	9824.4	4021306.7
EMQX	10000	1	string	352.139	325.4	98.312	230.1	3074.3	864.6	3594982.7
EMQX	10000	1	string	375.386	351.5	118.818	230.1	4802	780.7	3831831
EMQX	10000	1	string	368.113	344	107.195	227.5	4455.3	648.4	3754604.8
EMQX	10000	1	string	361.572	333.15	97.387	232.4	1988.9	834.8	3689083
EMQX	10000	2	binary	422.132	366.9	215.951	233.5	11953	11990.8	4368230.2
EMQX	10000	2	binary	397.056	339.6	169.571	233	4247.4	1687.3	4054661.3
EMQX	10000	2	binary	396.426	341.4	163.162	224.6	3486	983.9	4054491.4
EMQX	10000	2	binary	388.416	335.8	152.967	227.6	3360.8	704.9	3966865.3
EMQX	10000	2	binary	394.553	335.5	170.653	224.1	3176.8	966.4	4027237.1
EMQX	10000	2	json	456.767	378.8	349.42	232.7	23575.6	23607.7	4705552.4
EMQX	10000	2	json	417.257	347.35	215.787	230.2	3967.4	1417.8	4260553.4
EMQX	10000	2	json	405.898	349.9	173.484	237.2	3165	978.7	4147388.7
EMQX	10000	2	json	404.029	342.3	181.254	234.6	3967	1160.7	4124524.6
EMQX	10000	2	json	414.675	353.4	186.991	219.7	3195.4	844.3	4234267.4

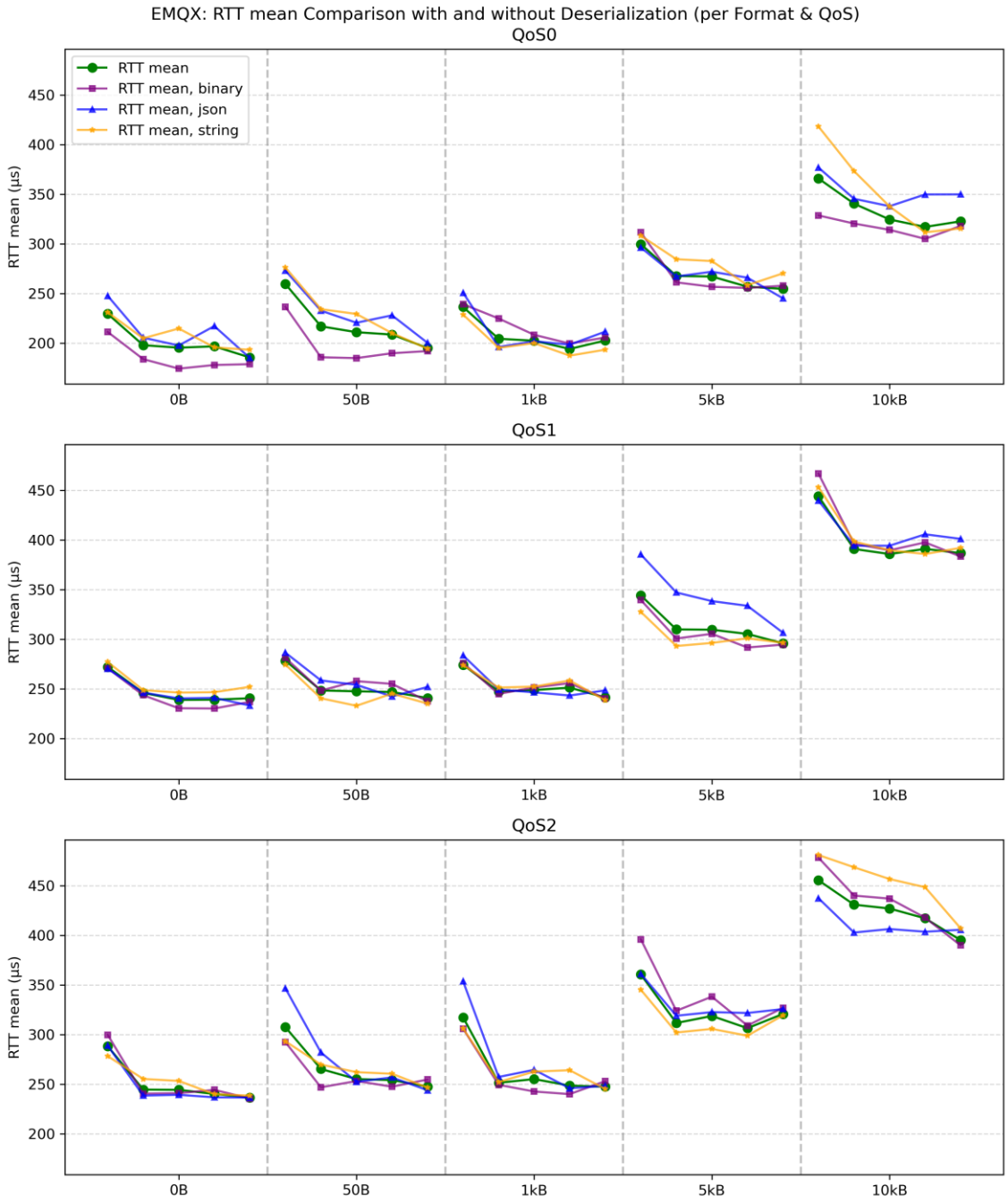
EMQX	10000	2	string	420.915	368.8	195.592	222.2	10220.8	10281.5	4349942.7
EMQX	10000	2	string	396.029	346.6	163.882	230.6	7439.6	596.7	4043867.9
EMQX	10000	2	string	383.748	336.5	144.604	219.5	5036.8	687	3917635.8
EMQX	10000	2	string	383.292	336	156.4	227.1	4256.8	778.3	3917183.4
EMQX	10000	2	string	391.969	343.6	137.728	236.4	1901.4	747.3	3999583
EMQX	100000	0	binary	1599.122	1492.4	456.835	1023.7	13375.5	13418.8	16137629.2
EMQX	100000	0	binary	1516.091	1427.7	379.964	970.7	6368.2	1920.7	15287776.7
EMQX	100000	0	binary	1502.84	1420.05	363.894	998.9	4978.6	1904.4	15152320.2
EMQX	100000	0	binary	1514.07	1422.7	362.667	996.3	5929.5	4873.8	15265728.7
EMQX	100000	0	binary	1471.42	1391.6	419.137	960.1	21269.1	2262.3	14829409.5
EMQX	100000	0	json	1749.07	1611.55	611.856	987.3	27100.3	27135.9	17634310.7
EMQX	100000	0	json	1712.832	1564.9	572.48	1005.6	10969.5	1962.9	17261634.2
EMQX	100000	0	json	1699.503	1572.9	548.291	1012.9	10140.4	2571.9	17121412
EMQX	100000	0	json	1690.92	1568.4	507.701	991	8718.5	2693.7	17033376
EMQX	100000	0	json	1680.493	1553.95	528.678	1008.8	9373.3	2762.2	16929329.3
EMQX	100000	0	string	1700.412	1581.9	545.185	969.3	13842.5	13879.9	17138337.1
EMQX	100000	0	string	1637.93	1522	512.215	969.9	9530.8	2207.2	16502173
EMQX	100000	0	string	1635.655	1507.4	585.327	957.8	20149	2143.1	16475495.1
EMQX	100000	0	string	1619.159	1502.6	513.624	942.9	8543.4	3422.5	16314468.3
EMQX	100000	0	string	1615.821	1499.25	520.92	957.8	8506.5	1973.5	16278433.2
EMQX	100000	1	binary	1893.673	1752.3	553.747	973.4	15434.3	15478.6	19114482
EMQX	100000	1	binary	1694.464	1589.7	501.201	1023.9	18057.1	2589.8	17078503.4
EMQX	100000	1	binary	1625.468	1522.2	390.983	1045.9	6935.8	2057	16386524.6
EMQX	100000	1	binary	1614.235	1519.15	391.68	984.6	8408.4	3450.3	16266161.3
EMQX	100000	1	binary	1599.043	1502.25	383.456	984.9	8316.4	2363.6	16114181
EMQX	100000	1	json	1824.938	1693.45	604.195	1072.6	32020.1	32063.2	18413578.6
EMQX	100000	1	json	1688.298	1586.6	433.991	984.7	7693.8	2196.9	17011610.7
EMQX	100000	1	json	1697.576	1584.9	465.745	1027.8	8998.8	2279.4	17104288.4
EMQX	100000	1	json	1684.307	1565.05	478.662	1071	9502.3	2368.3	16970031.5
EMQX	100000	1	json	1679.956	1571.85	446.067	1008.1	7747.3	1753.8	16925935.4
EMQX	100000	1	string	1991.338	1864.55	577.422	1122.4	13681.6	13728.9	20090768.2
EMQX	100000	1	string	1874.165	1753.6	543.332	1133.7	11028.4	3318.3	18888298.3
EMQX	100000	1	string	1735.936	1634.4	446.189	1012.7	6631.2	2776.1	17502565
EMQX	100000	1	string	1710.566	1604.15	444.637	1069.9	8710.7	4043.9	17240559.9
EMQX	100000	1	string	1755.549	1658.7	437.465	1078.7	6171.6	2818.9	17689136.1
EMQX	100000	2	binary	1712.121	1593.65	451.847	1040.5	12764.7	12814.4	17274320.9
EMQX	100000	2	binary	1684.496	1594.8	412.017	1001.8	4734	2088.9	16981448.1
EMQX	100000	2	binary	1663.472	1565.85	387.899	1016.4	4839.1	2504.3	16769342
EMQX	100000	2	binary	1650.054	1552.25	398.82	1008	6770.6	1885.3	16637454.2
EMQX	100000	2	binary	1664.182	1567.25	430.157	1034.3	11151.5	2414.7	16774884.1
EMQX	100000	2	json	1806.301	1671	592.829	1056.8	27829.9	27864	18222466.2
EMQX	100000	2	json	1750.482	1631.7	500.828	1054.3	9908.8	2395	17638711.2
EMQX	100000	2	json	1735.75	1605.7	501.607	1065	8442.3	2331.2	17490833.3
EMQX	100000	2	json	1751.798	1643.3	463.727	1052.3	6177.5	2707.2	17657890.2
EMQX	100000	2	json	1768.088	1644.65	484.545	1057.6	6786	1996	17817014.1
EMQX	100000	2	string	1988.294	1852.95	581.118	1057.4	13938.7	13981.4	20057760.4

EMQX	100000	2	string	1759.684	1654.05	461.853	1044	6031.6	2551	17736788.8
EMQX	100000	2	string	1727.207	1628.15	446.076	1025.8	6120.3	3033	17408515.7
EMQX	100000	2	string	1713.485	1607.1	468.55	1057.1	16608.6	2558.4	17267848.8
EMQX	100000	2	string	1773.86	1669.1	451.998	1103.9	6358.5	2989.8	17876544.1

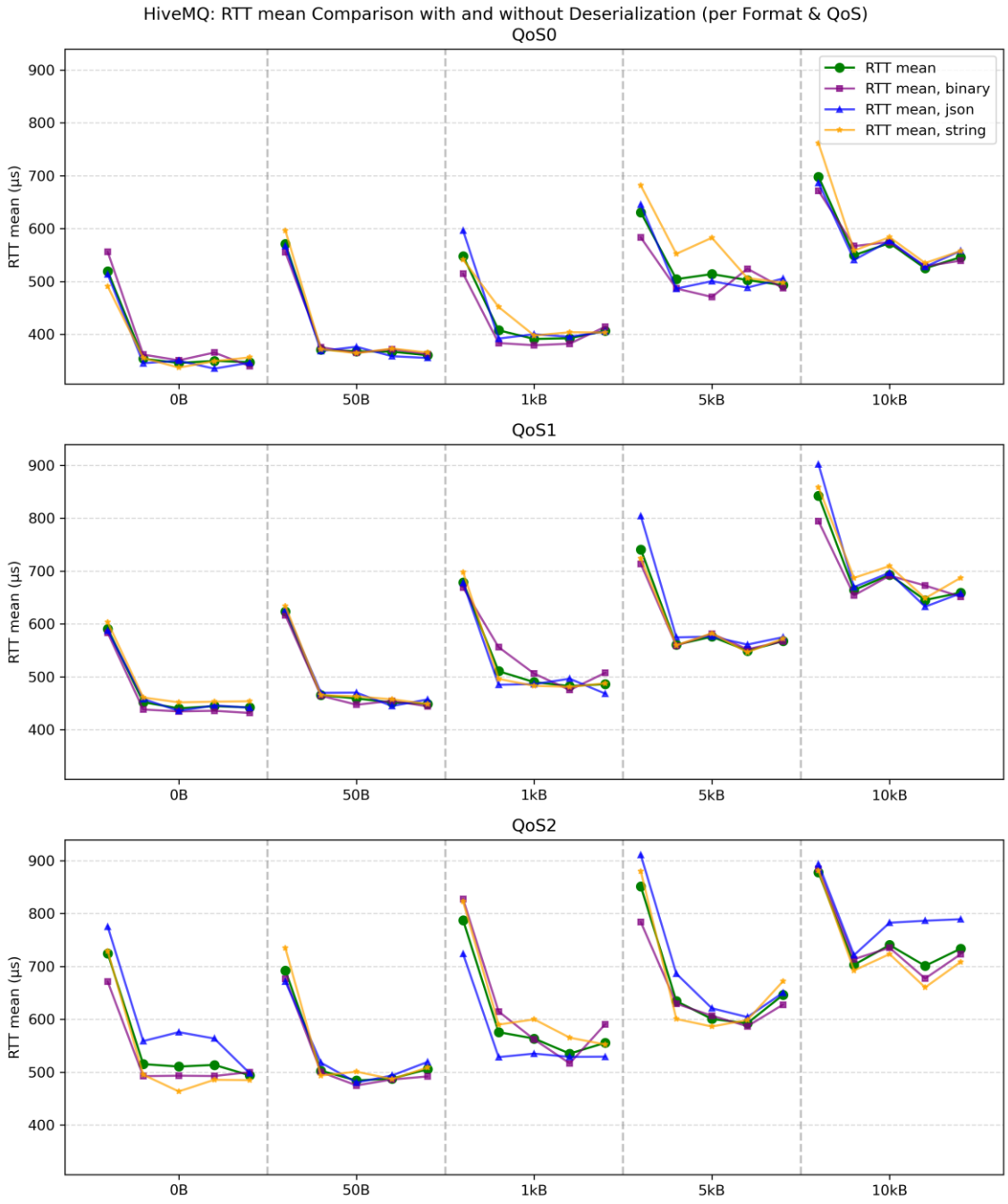
Appendix 2. Raw measurements (NanoMQ, 0b, QoS0, binary)



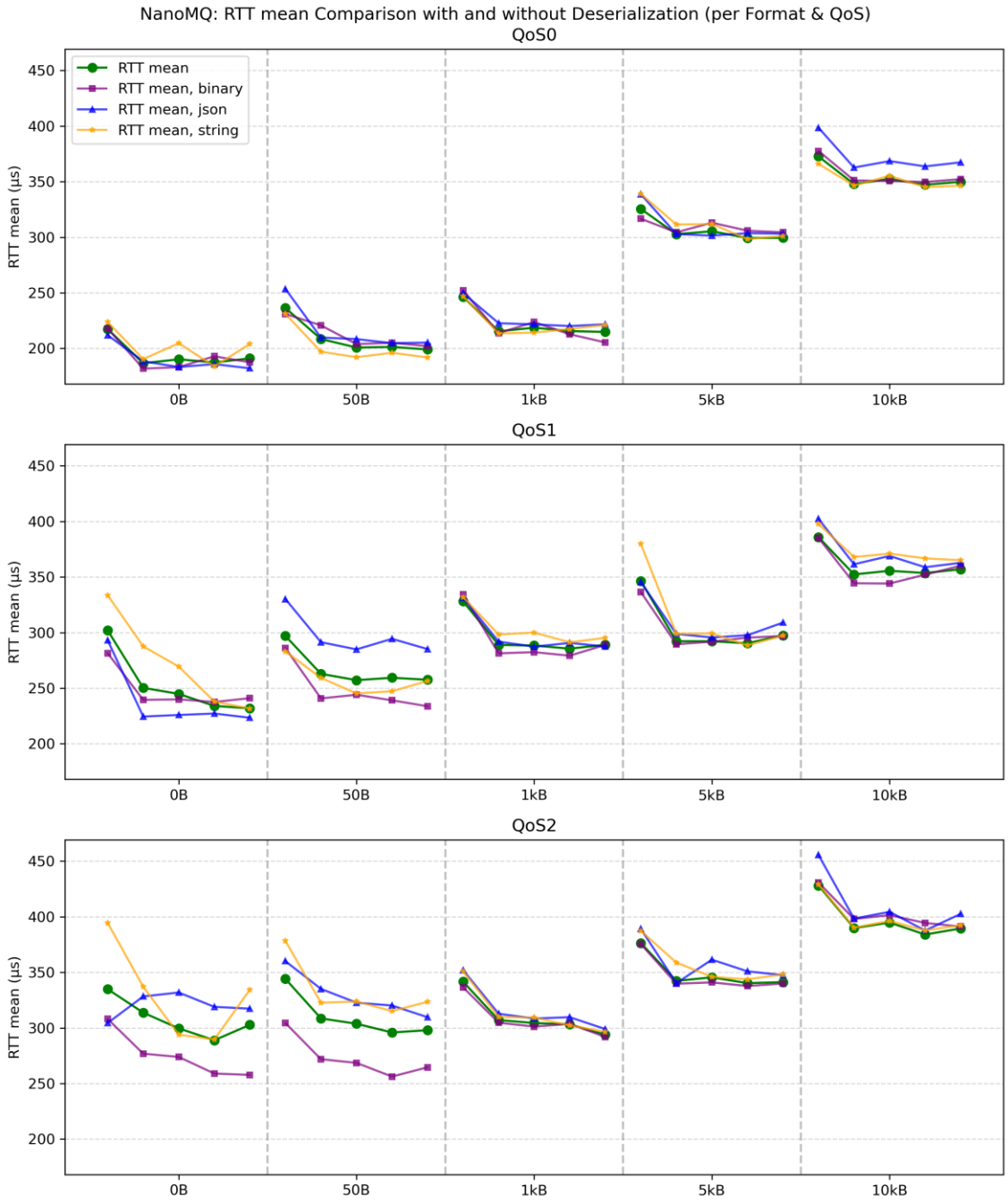
Appendix 3. Emqx RTT mean with and without deserializing time



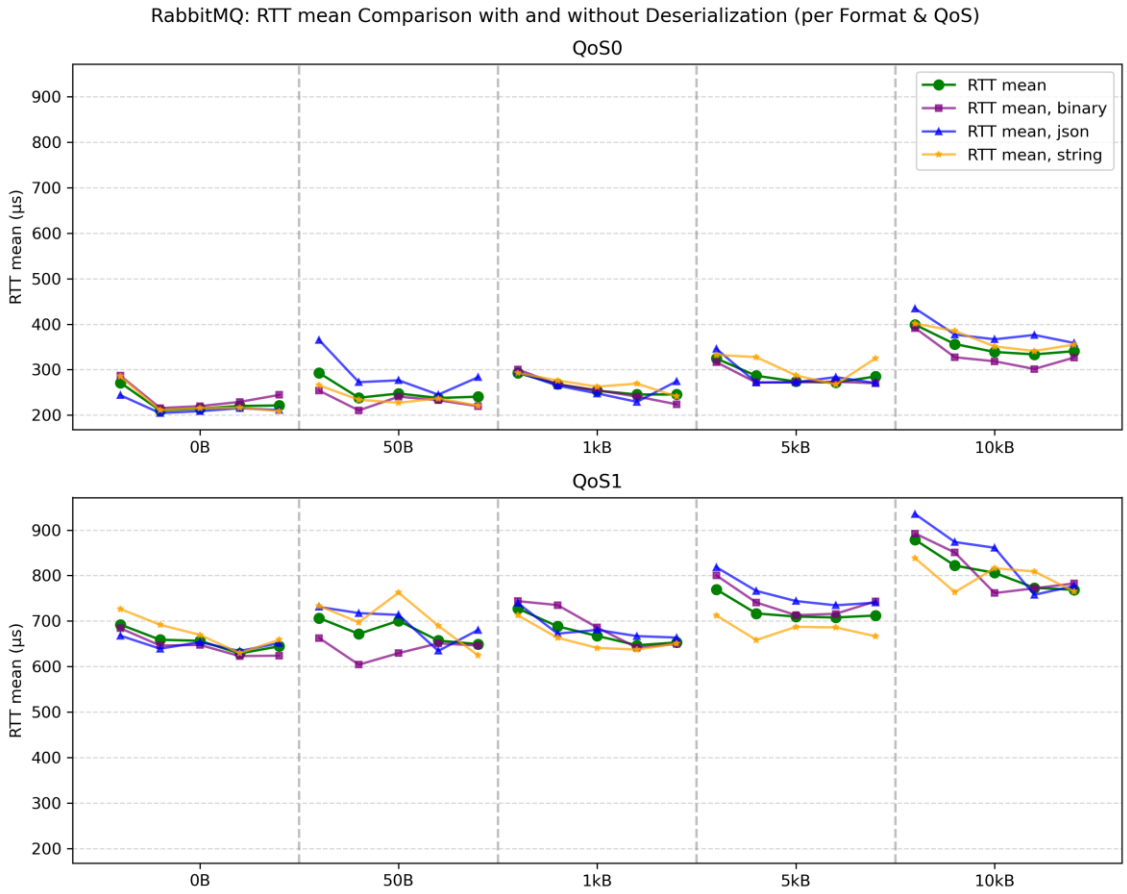
Appendix 4. HiveMQ RTT mean with and without deserializing time



Appendix 5. NanoMQ RTT mean with and without deserializing time



Appendix 6. RabbitMQ RTT mean with and without deserializing time



Appendix 7. RTT median per broker, payload size and QoS with 1ms line.