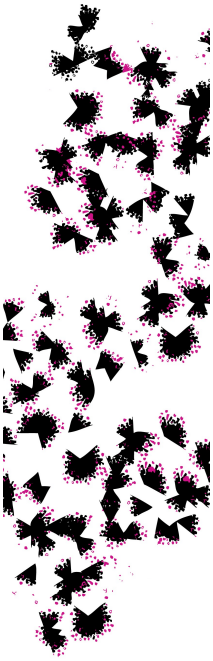

Using the Object Management Group Data Distribution Service to reliably teleoperate robotic systems

R.H. (Roland) Roelofs

MSc Report

Committee:
Prof.dr.ir. G.J.M. Krijnen
K.J. Russcher, MSc
Dr.ing. D.M. Ziener

August 2018

014RAM2018
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

MIRA CTIT
BIOMEDICAL TECHNOLOGY AND TECHNICAL MEDICINE

Summary

The purpose of this thesis is to investigate whether or not Data Distribution Service (DDS) from the Object Management Group (OMG) can be used to reliably teleoperate (operate over a wireless network) robotic systems used by the National Police of the Netherlands (NPN). For the purpose of this research, 'reliably' refers to having a high ($\simeq 100\%$) probability to meet the latency requirements under specified operating conditions. These operating conditions include different percentages of *packet loss*, *network overhead* and different kinds of *Operating Systems (OSs) and hardware*. In this research, each operating condition is subjected to different amounts of data (ranging from 4 bytes to 16,384 bytes) and different DDS Quality of Service (QoS) settings (with DDS QoS Reliability set to 'reliable' and 'best effort'). To date, a limited number of tests have been conducted to validate latency when transferring data using DDS. This thesis extends current research in order to draw a conclusion regarding whether DDS can be used to reliably teleoperate robotic systems used by the NPN. These results are also applicable to other systems with similar latency requirements.

The police tasks investigated in this study are categorized as *impressive* or *less impressive* and then rated according to latency requirements: **less impressive acceptable (200 ms)**, **impressive acceptable (75 ms)**, **less impressive preferred (50 ms)**, and **impressive preferred (25 ms)**.

The robotic systems are usually teleoperated over a wireless network. However, for testing a wired network is used, since they are less influenced by the environment. The latency of a wired network is 5 ms to 75 ms lower than that of a wireless network, so latency requirements have to be scaled accordingly for the wired network used in this study. The wired network latency requirements for each police task are as follows: **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)**.

In order to validate whether or not DDS can be used to reliably teleoperate robotic systems a test setup is created. The test setup measures latency by sending data between two devices while separately inducing different operating conditions. The operating conditions used in this study are 0%, 8%, and 16% packet loss; 0%, 50%, and 90-99% network overhead; OSs Windows 10, Ubuntu 16.04, and Raspbian (using laptops and a Raspberry Pi 3).

Validating DDS while adjusting the operating condition *packet loss* (Table 1 first row) indicates that DDS cannot be used to reliably teleoperate robotic systems used by the NPN when there is an 8% or higher packet loss with DDS QoS Reliability set to either 'reliable' or 'best effort' for impressive and less impressive tasks. However, for lower packet loss percentages DDS can be used, since both sample received percentage and meeting the impressive and less impressive latency requirements increases up to 100%.

Validating DDS while adjusting the operating condition *network overhead* (Table 1 second row) indicates that DDS can be used to reliably teleoperate robotic systems used by the NPN for less impressive tasks when the used bandwidth is less than the maximum bandwidth. This also applies for impressive tasks with a data size of up to 4,096 bytes. DDS cannot be used to reliably teleoperate robotic systems used by the NPN when the used bandwidth exceeds the maximum bandwidth.

Validating DDS while adjusting the operating conditions *OS and hardware* (Table 1 third row) indicates that DDS can be used to reliably teleoperate robotic systems used by the NPN with OS Ubuntu 16.04 and Raspbian in most situations. The only situation in which DDS cannot be used to reliably teleoperate robotic systems is for impressive tasks when distributing data with a size of 16,384 bytes between OS Ubuntu 16.04 and OS Raspbian. In addition, DDS cannot be used to reliably teleoperate robotic systems used by the NPN when one of the OSs is Windows 10, except for less impressive tasks. Resource-rich hardware and resource-restricted hardware can both be used to reliably teleoperate robotic systems. It is recommended to use resource-rich hardware over resource-restricted hardware, since it has a lower latency median and higher probability of meeting the latency requirements.

Operating Conditions		Imp. Pref.	Less Imp. Pref.	Imp. Accept.	Less Imp. Accept.
Packet Loss	0%	yes (if data size ≤ 4,096 B)			yes
	8%	no	no	no	no
	16%	no	no	no	no
Network Overhead	Used Bandwidth < Max. Bandwidth	yes (if data size ≤ 4,096 B)			yes
	Used Bandwidth ≥ Max. Bandwidth	no	no	no	no
OS and Hardware	Ubuntu 16.04 ⟷ Ubuntu 16.04	yes	yes	yes	yes
	Ubuntu 16.04 ⟷ Raspbian	yes (if data size ≤ 4,096 B)			yes
	Windows 10 ⟷ Ubuntu 16.04	no	no	yes (≤ 1,024 B)	yes
	Windows 10 ⟷ Raspbian	no	no	no	yes (≤ 4,096 B)

Table 1: Whether DDS can be used to reliably teleoperate robotic systems used by the NPN for impressive (Imp.) and less impressive (Less Imp.) preferred (Pref.) and acceptable (Accept.) tasks when subjected to different operating conditions (packet loss, bandwidths, OSs and hardware) are shown in the white area.

The findings of this study suggest that DDS can be used to reliably teleoperate robotic systems used by the NPN when there is; few (≈ 2%) to no packet loss, used bandwidth is below the maximum bandwidth, and one of the validated OSs except Windows 10 (in most cases) is used.

Contents

1 Introduction **1**

1.1 Context . . . 1

1.2 Research Questions . . . 2

1.3 Approach . . . 2

1.4 Outline . . . 3

2 Background Information **4**

2.1 Reliability . . . 4

2.2 Latency Requirements . . . 5

2.3 Data Distribution Service . . . 9

2.4 Operating Conditions . . . 10

2.5 Conclusion . . . 12

3 Method **13**

3.1 Test Setup . . . 13

3.2 Conversion . . . 17

4 Results **21**

4.1 Research Sub-Question I: Packet Loss . . . 21

4.2 Research Sub-Question II: Network Overhead . . . 23

4.3 Research Sub-Question III: Operating Systems and Hardware . . . 25

5 Discussion **28**

5.1 Review of the Findings . . . 28

5.2 Other Findings . . . 31

5.3 Conclusion . . . 32

5.4 Limitations . . . 33

5.5 Recommendations . . . 33

A Results **35**

A.1 Baseline Results . . . 35

A.2 Packet Loss Results . . . 36

A.3 Network Overhead Results . . . 37

A.4 Operating Systems and Hardware Results . . . 38

A.5 Latency Spikes Results . . . 39

B User Manual **42**

B.1 Automatic Testing . . . 42

B.2 Manually Testing 43

C Compiling Testbench **45**

Chapter 1

Introduction

1.1 Context

These days Data Distribution Service (DDS) [13] is used in many modern-day applications. DDS is a data-centric middleware protocol created by the Object Management Group (OMG) to distribute data. According to their site [8], DDS provides "low-latency, extreme reliability, and a scalable architecture" by integrating the components of a system together. Other advantages of DDS include ease of integration, advanced security, Open Standards, rich set of Quality of Service (QoS), and compatibility with many systems [8].

This research is part of the project Robots voor Veiligheid (Robots for Security, RoVe) [16], which also uses DDS for data distribution. Project RoVe is a collaboration between the National Police of the Netherlands (NPN) and the Robotics and Mechatronics (RaM) group. The NPN uses robotic systems for tasks such as observation and surveillance. For example, police officers might use a drone to create 3D-images from a crime scene or accident [1]. For these tasks, the robotic system needs to be controlled reliably. One of the factors that influences the reliability of this control is latency. Latency, which is the main focus of this research thesis, refers to the amount of time between when data is sent and when it is received, as shown in Figure 1.1. Other factors such as data loss and data corruption also influence reliability, but are not the focus of this research thesis. However data loss is used to validate the effects on latency.

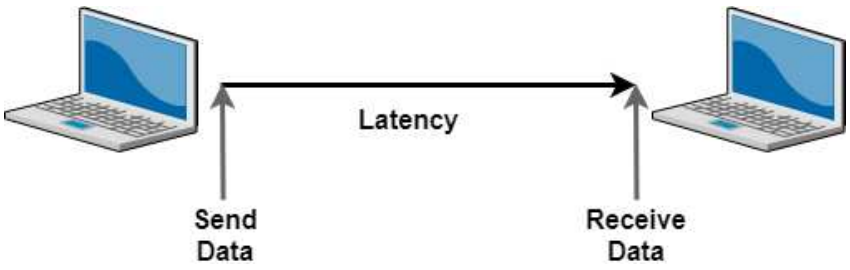

Figure 1.1: Latency is defined as the amount of time between when data is sent and when it is received.

To date, a limited number of tests have been conducted to validate latency when transferring data using DDS. Previous studies have mainly compared DDS with other types of data distribution. For instance, one study [18] compared the latency of DDS with the traditional socket-based solution for communication. Another study [3] analyzed the latency difference between PrimsTech and Real-Time Innovations (RTI), which are the two main stakeholders that have implemented DDS in the DDS market. A literature search found no studies that validated latency while inducing packet loss or network overhead [12] and using different Operating Systems (OSs) and hardware. These parameters also influence latency, as described in Section 2.4.

1.2 Research Questions

This research was conducted in order to draw a conclusion as to whether or not DDS can be used to reliably teleoperate robotic systems used by the NPN. This research will build upon current research by validating the latency requirements needed to reliably teleoperate robotic systems under different conditions, including different percentages of packet loss, network overhead, and different OSs and hardware (Windows 10 and Ubuntu 16.04). DDS was then subjected to different data sizes (ranging from 4 bytes to 16,384 bytes) and different types of DDS QoS settings (DDS QoS Reliability settings of 'reliable' and 'best effort').

The main research question is as follows:

- Can DDS be used to reliably teleoperate robotic systems used by the NPN under specified operating conditions?

Following from this, the main research question is divided into three sub-questions:

- Does DDS meet the latency requirements when subjected to different data sizes (ranging from 4 to 16,384 bytes) and different types of DDS QoS settings (DDS QoS Reliability settings of 'reliable' and 'best effort') used under the following operating conditions:
  1. packet loss,
  2. network overhead,
  3. different OSs and hardware.

1.3 Approach

In order to validate whether DDS can be used to reliably teleoperate robotic systems, a test setup was created. The test setup measured the latency by sending data from the master computer to slave computer and back, as shown in Figure 1.2. The latency was computed by subtracting the 'send data' time from the 'receive data copy' time and then dividing it by two. However, this is a simplified version of what was done and it will be explained later in Section 3.1.1.

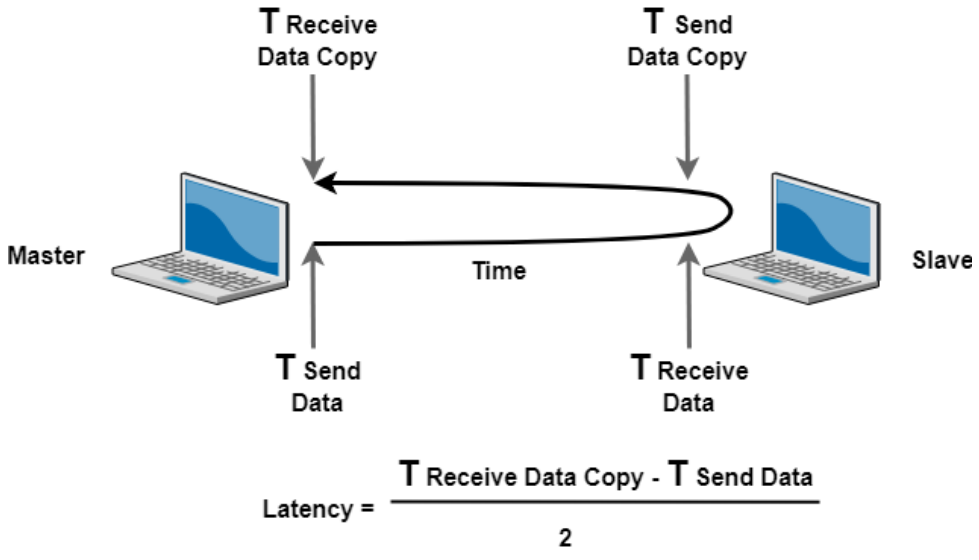

Figure 1.2: Abstract overview of the setup used to measure the latency during tests.

The tests were performed over a wired network in order to obtain reliable and repeatable results. A wired network was used because a wireless network is more sensitive to noise than a wired network. The results were then converted to corresponding values for a wireless network in order to validate the main research question of whether DDS can be used to reliably teleoperate robotic systems used by the NPN under specified operating conditions. This conversion rate was calculated by comparing two baseline tests—one baseline test using a wireless network and one baseline test using a wired network. The difference was then applied to the wired network test results in this study, which lowered the latency requirement.

For the first research sub-question, a specified percentage of the total transmitted data was dropped using the program qdisc to simulate packet loss. For the second sub-question, additional data was transmitted to simulate network overhead. For the third sub-question, different OSs, including Windows 10, Ubuntu 16.04, and Raspbian, as well as different hardware including a laptop and a Raspberry Pi 3, were used.

The validation of each research question was also subjected to varying data sizes ranging from 4 to 16,384 bytes, which was the range used in the previous studies mentioned earlier. For the DDS QoS Reliability, the settings 'reliable' and 'best effort' were used. The tests were performed on a resource-rich device (laptop) and resource-restricted hardware (single board computer). These hardware choices correspond to the hardware used by the NPN. Each operating condition was efficiently and thoroughly tested by using 10,000 samples sent at a frequency of 50 Hz.

1.4 Outline

The second chapter of this thesis will begin by explaining what reliability is and what the latency requirements are to reliably teleoperate robotic systems used by the NPN. Background information on DDS will be discussed, along with the operating conditions of packet loss, network overhead, and OS and hardware. Chapter 3 will address the method used to validate whether DDS can be used to reliably teleoperate robotic systems used by the NPN, and the results of this study will be stated in Chapter 4. A discussion of these results in regards to latency requirements, other findings, conclusions, and recommendations is found in Chapter 5.

Additional tables and figures are presented in Appendix A. The user manual for running the tests is presented in Appendix B, and the manual to compile the code is presented in Appendix C. At the time of writing, the code is property of the project Robots voor Veiligheid (Robots for Security, RoVe) and located in the Robotics and Mechatronics (RaM) git. For more information, contact Klaas Jan Russcher at the University of Twente.

Chapter 2

Background Information

In order to provide a background for this research, the term 'reliability' is explained in Section 2.1. The latency requirements to reliably teleoperate robotic systems used by the National Police of the Netherlands (NPN) are addressed in Section 2.2. Data Distribution Service (DDS) is described in Section 2.3, which also addresses the parts used during the performance validation. Finally, Section 2.4 explains the parameters of packet loss and network overhead [12], which influence the performance of a wireless network and were chosen to validate the performance of DDS.

2.1 Reliability

The term 'reliability' can have different meanings. Therefore, the definition with respect to this project will be addressed first. Reliability is, according to Blanchard and Fabrycky (2014) [5], defined as "the *probability* that a system will accomplish a *designated mission* in a *satisfactory manner* under specified *operating conditions*". This definition consists of a mission and three important elements: probability, satisfactory performance, and operating conditions.

The *designated mission* is split into a high-level and a low-level mission. The focus of this project lies with the latter. The high-level mission is to successfully carry out an operation, such as surveillance or observation, using a robotic system. The low-level mission is to receive data from a robotic system and controller within a specified time period. The low-level mission is a part of the high-level mission, but failures of the two missions are not linked. In other words, a high-level mission failure can still occur despite low-level mission success, and vice versa.

Probability is expressed as a fraction or percentage indicating the number of times the *designated mission* will succeed over a total number of attempts [5]. The reason that reliability is described in probabilistic terms is because failures inevitably occur at some point in time. For example, the probability of control signals being received by a robotic system is 75%, meaning that 75% of the control signals are being received within a given period of time. The probability for this research refers to the number of data samples received within a given period of time, and 'within a given period of time' refers to meeting the latency requirements described in Section 2.2.2.

The term *satisfactory manner* requires a definition of what is considered to be satisfactory [5]. Satisfactory performance for the low-level mission is that the latency of data samples complies with the latency requirement. The high-level mission can consist of multiple tasks with different complexities or difficulties. For this study, the difficulty of a task is categorized as 'impressive' or 'less impressive', each with a different latency requirement. This categorization is addressed in Section 2.2.1.

Operating conditions specify the conditions under which a mission is expected to operate [5]. These conditions refer to the network factors under which a system is operating. For this research, the operating conditions encompass different percentages of packet loss and different amounts of network overhead, as well as different Operating Systems (OSs) and hardware, all of which are validated against different data sizes and DDS Quality of Service (QoS) Reliability. DDS QoS Reliability is addressed in Section 2.3.1, and the operating conditions of packet loss and network overhead are further explained in Section 2.4.

2.2 Latency Requirements

Satisfactory performance, or performance that meets latency requirements, of high-level and low-level missions is discussed in this section. As stated in the previous section, a high-level mission is to successfully carry out an operation using a robotic system. These operations consist of multiple tasks. For this research, the complexity of these tasks is divided into 'impressive' and 'less impressive', as discussed in Section 2.2.1. Each category of complexity has different preferred and acceptable latency requirements, as visualized in Figure 2.1. The acceptable and preferred latency requirements summarized in Section 2.2.2 are generalized and subject to the effects of several moderating factors including display characteristics, viewing conditions, additional cues, and user experience, and should therefore be used as a guideline rather than a rule.

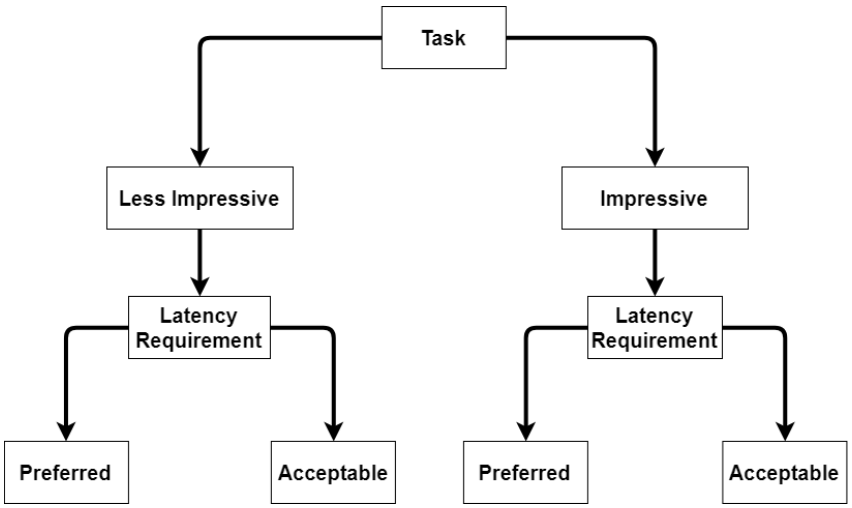

Figure 2.1: The complexity of a task is divided into the categories 'impressive' and 'less impressive'. Both have an acceptable latency requirement and a preferred latency requirement.

2.2.1 Tasks

As stated before, the complexity of tasks for this study have been divided into 'impressive' and 'less impressive'. The idea to divide tasks into 'impressive' and 'less impressive' comes from the paper published by Chen and Thropp (2007) [6]. Both complexities have a preferred and an acceptable latency requirement, as visualized in Figure 2.1.

The complexity of a task is dependent on two factors:

- the relative *speed* at which a robotic system and surrounding object(s) are traveling towards each other, and
- the relative *distance* between a robotic system and an object(s).

The distance over which teleoperation takes place is not included, since it indirectly affects the complexity via latency (assuming each robotic system has onboard cameras, thus have visual of the surroundings nearby the robotic system). Latency is for this study not a factor that influences the complexity, but a requirement in order to perform the impressive and less impressive tasks in a satisfactory manner.

The combined relative speed and distance determines the complexity of a task. The complexity designations of 'impressive' or 'less impressive' are qualitative and based on a sliding scale. This scale is indicated by the dark-to-light gradient illustrated in Figure 2.2. Tasks characterized by a relatively high speed and a short distance between the robotic system and object are considered to be impressive, whereas tasks with a relatively low speed and a large distance between the robotic system and object are considered to be less impressive.

During missions, tasks can shift from 'impressive' to 'less impressive', and vice versa. In one example, a police officer is teleoperating a robotic system at a relatively high speed inside a mall where many people are walking around. At first, the distance between the robotic system and the people walking

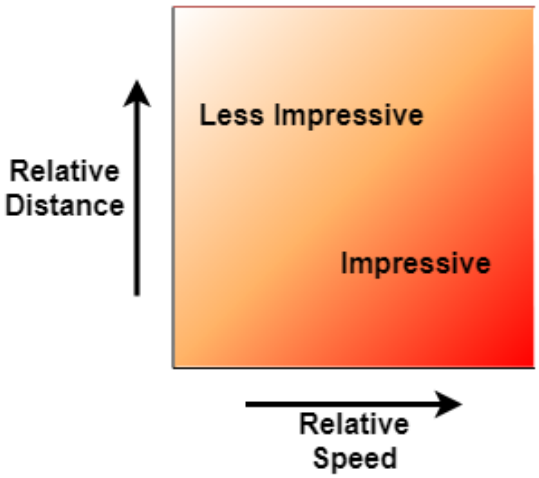

Figure 2.2: The complexity of a task is dependent on the relative distance and speed between a robotic system and an object(s). The gradient from dark to light and the absence of numeric values is to indicate that the line between 'impressive' and 'less impressive' tasks is qualitative rather than quantitative.

around is relative small. This would make the task impressive due to the relatively high speed of the robotic system. Upon entering a different section of the mall with no people around and a relatively large distance between the robotic system and other objects, the impressive task changes to a less impressive task. Returning to the crowded part of the mall at a relatively high speed where many people are walking around at close range makes the task impressive again.

Impressive Tasks

Teleoperating robotic systems during impressive tasks can be compared to video gaming. An impressive task can be compared to for instance a racing game. In these games, cars manoeuvre around other cars and objects in high velocity, requiring a short reaction time. The needed short reaction time combined with the small distance to other object and the fast pace are characteristics of an impressive task.

The ability to react quickly is not only dependent on the user's reaction time, but also on the system's performance. If the system's performance does not comply with the latency requirements, gameplay will be negatively affected. For example, inside a race game, a car is being controlled driving through a town at a relatively high speed and with a relatively small distance between it and other objects. In this case, a late stop signal could result in the car crashing into another object. Multiple crashes decrease the user's chances of winning the game, or completing the high-level mission. The same concept applies to teleoperating robotic systems during impressive tasks. If the system performance, or low-level mission, does not comply with latency requirements, the high-level mission could fail.

In conclusion, the complexity of both playing fast-paced video games and teleoperating robotic systems are impressive. Therefore, the latency requirements of fast-paced video games also apply to teleoperating robotic systems used by the NPN.

The acceptable latency requirements are between 50 ms and 150 ms [10]. The preferred latency requirement is a maximum of 50 ms. At this latency, users achieve optimum performance.

Less Impressive Tasks

Teleoperating robotic systems during less impressive tasks can be compared to playing a slow-paced video game such as The Sims. Scenarios are slow paced, and high-level missions are achievable with a longer response time compared to impressive tasks or fast-paced video games. This is because the game is more affected by strategy than technical performance. Less impressive tasks have a relatively

large distance between objects and/or a lower speed, which allows for a higher response time compared to impressive tasks.

The acceptable latency requirement for slow-paced video games is between 100 ms and 400 ms [7]. The preferred latency requirement is a maximum of 100 ms [17]. This latency provides the best user experience. Because of the similarities in gaming and teleoperating, it is assumed that these values hold for the purposes in this research. Therefore, the values mentioned above are chosen to be the acceptable and preferred values for this research.

Latency Measurement

The acceptable and preferred latency requirements for impressive and less impressive tasks, addressed in the previous Subsections, are measured using a ping test or a similar technique. Ping is a build-in program available on most OSs used to measure latencies between the client and the server. The latency measurement shown in Figure 2.3 begin when the client transmits data to the server. Once the server receives this data, it immediately sends it back to the client. The measurement stops once the client receives the data. The time between when the client transmits the data and when the client receives the data is referred to as the latency, or round-trip time (RTT). It is important that the execution time, or the time between when the server receives the data and when the server transmits the data, is not neglected by assuming that complex computations are not required to send a signal back. This means that the execution time is also not part of the latency requirements from the studies.

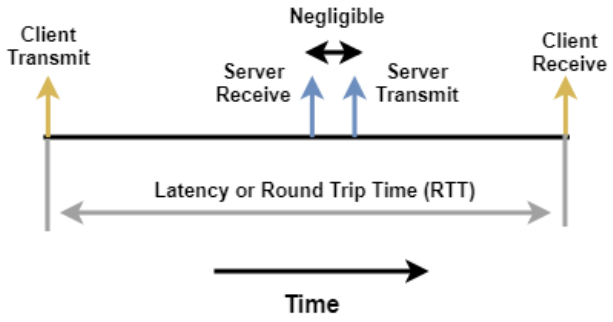

Figure 2.3: Time flow of ping or round-trip time (RTT).

For this research, the controller is the client and the robotic system is the server. As shown in Figure 2.4, and corresponding timing diagram in Figure 2.5, the latency requirements (divided by two) can be directly applied to the latency between the send and receive times, as the execution time is not part of the latency.

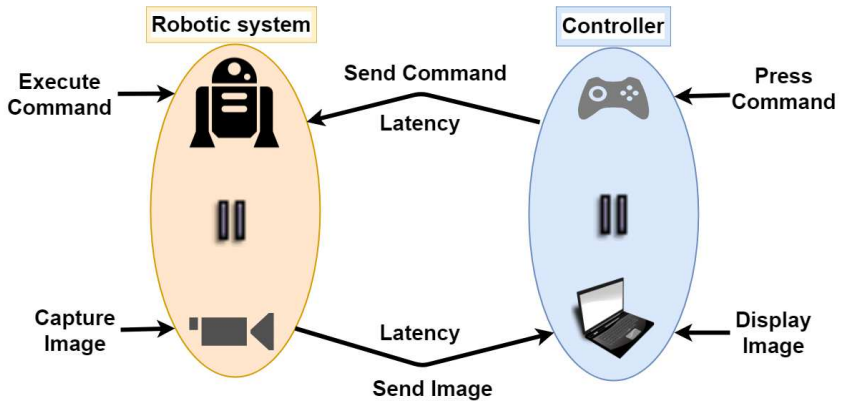

Figure 2.4: Data flow between the robotic system and the controller.

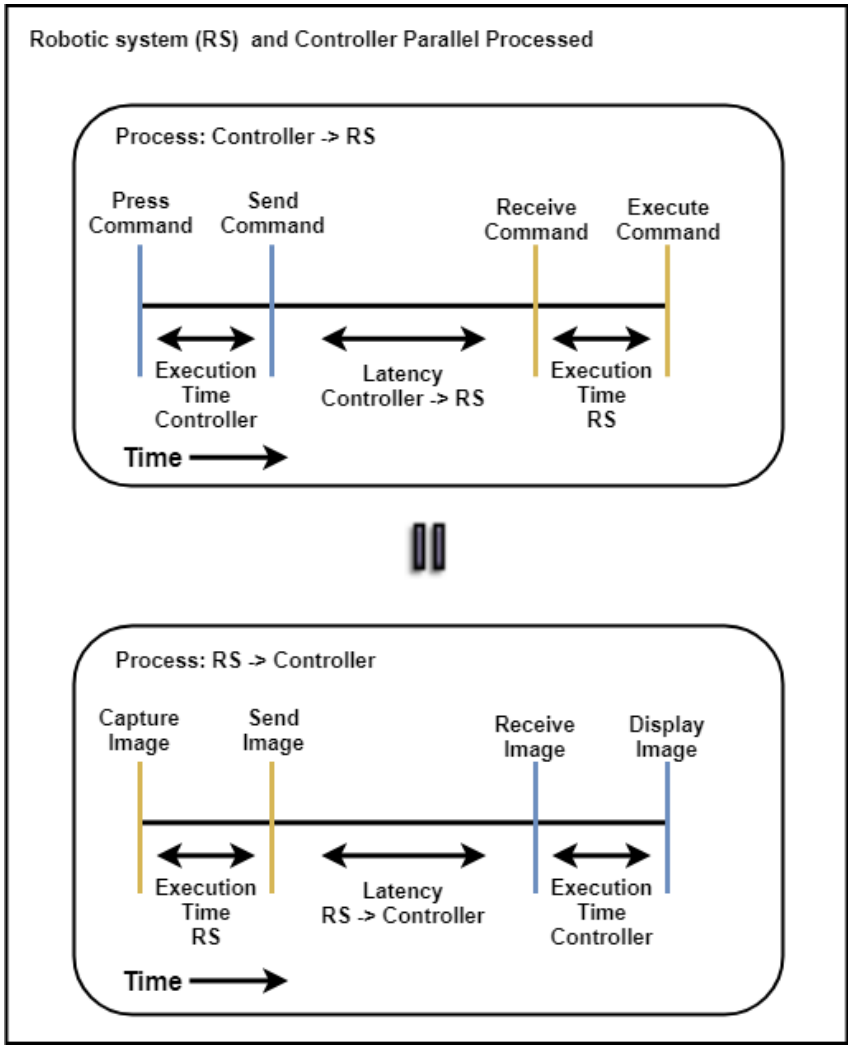

Figure 2.5: Timing diagram between the robotic system and controller.

2.2.2 Conclusion

The acceptable and preferred latency requirements for impressive and less impressive tasks (divided by two) are summarized in Table 2.1. These requirements are based on fast-paced and slow-paced video gaming, which are comparable to impressive and less impressive teleoperating tasks respectively. When these requirements are met, the probability of success of the high-level mission increases. The high-level mission is to successfully carry out an operation such as surveillance or observation using a robotic system. Based on the latency requirements and results stated in Chapter 4, the research sub-questions are discussed and answered in Chapter 5. The purpose of this research is to determine whether DDS can be used to reliably teleoperate robotic systems under specified operating conditions, with 'reliably' referring to a high ($\simeq 100\%$) probability of meeting latency requirements.

	Latency (ms)	
Task	Acceptable	Preferred
Less impressive	200	50
Impressive	75	25

Table 2.1: The acceptable and preferred latency requirements (in milliseconds) to reliably teleoperate robotic systems used by the NPN for impressive and less impressive tasks.

2.3 Data Distribution Service

As mentioned in the introduction, DDS [13] is validated whether or not it can be used to reliably teleoperate robotic systems used by the NPN. DDS is a publish—subscribe service that transfers data samples through the system as conceptual 'data objects', as shown in Figure 2.6 [14]. The publisher sends (or publishes) data samples to one or more subscribers. A subscriber can only subscribe to a publisher when both topics (the data object) have the same name, data type, and DDS QoS policies.

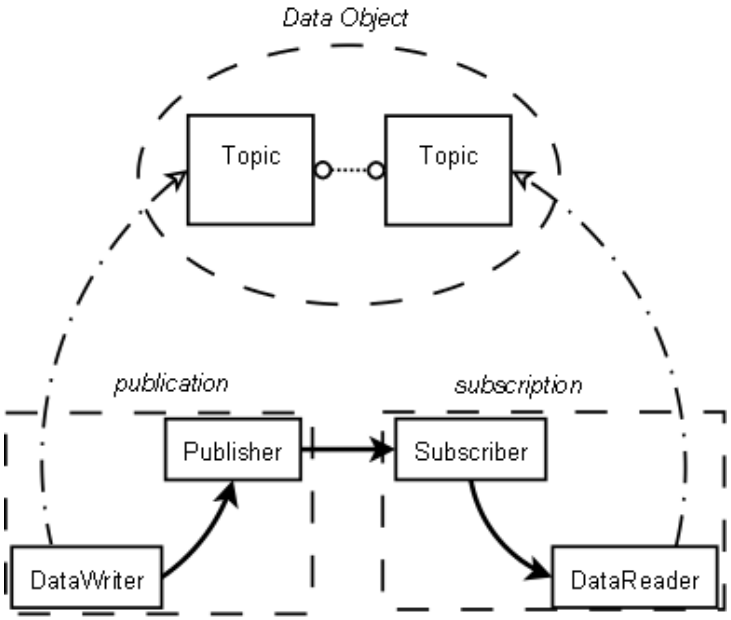

Figure 2.6: Simple conceptual overview of DDS showing how the basic components are interconnected with one another [14].

DDS consists of five basic components—topic, publisher, subscriber, dataWriter, and dataReader—as shown in Figure 2.6, and can be described as follows:

- Topics are the data objects that contain information about the name, data type, and QoS policies, and are used to make an association between publication and subscription.
- Publishers restrict and control the data that is send by the dataWriter(s).
- Subscribers restrict and control the data that is received by the dataReader(s).
- DataWriters send data of a data type corresponding to the topic's data type.
- DataReaders receive data of a data type corresponding to the topic's data type.

The data flow starts at the publication, where data is written to the dataWriter and published by the publisher. The published data is send to the subscriber with which it has a subscription. Finally, the subscriber passes the received data to the dataReader enabling the application to use its data.

2.3.1 DDS Quality of Service

DDS defines a set of QoS policies [13] that control the flow of data through the system. Topics, dataReaders, dataWriters, publishers, and subscribers all have QoS policies. The two QoS policies that can influence the system's latency and reliability are QoS Reliability and QoS Transport Priority.

DDS QoS Reliability is set either to 'reliable' or 'best effort'. DDS QoS Reliability 'best effort' is comparable to User Datagram Protocol (UDP) and DDS QoS Reliability 'reliable' is comparable to Transmission Control Protocol (TCP). When data or packets are lost, DDS QoS Reliability 'reliable' resends the lost data, resulting in an increased latency. DDS QoS Reliability 'best effort' does not resend lost data.

DDS QoS Transport Priority prioritizes sending more important data over sending less important data. The importance of data is indicated by an integer value, where a higher value indicates a higher priority. This DDS QoS policy has been implemented in the test setup, but the effect on latency has not been

validated due to time constraints and that it was discovered after making the plan of attack, therefore had a lower priority.

2.3.2 OpenDDS

This study used OpenDDS to validate whether DDS can be used to reliably teleoperate robotic systems used by the NPN. OpenDDS [14] is a free, open source C++ implementation of DDS used by the project Robots voor Veiligheid (Robots for Security, RoVe) at the time of writing. OpenDDS uses all the components of DDS as well as one additional component, namely DCPSInfoRepo, as illustrated in Figure 2.7. The DCPSInfoRepo is a specific object that maintains the state of the domain(s). It detects when subscriptions and publications in a domain should be associated and notifies them to make the associations. OpenDDS is not fully equivalent to DDS, but the tests for this research were minimally affected by the difference because the DCPSInfoRepo affects the start-up and not the latency of data transfer. This means that the results produced by tests using OpenDDS were applicable to the research sub-questions.

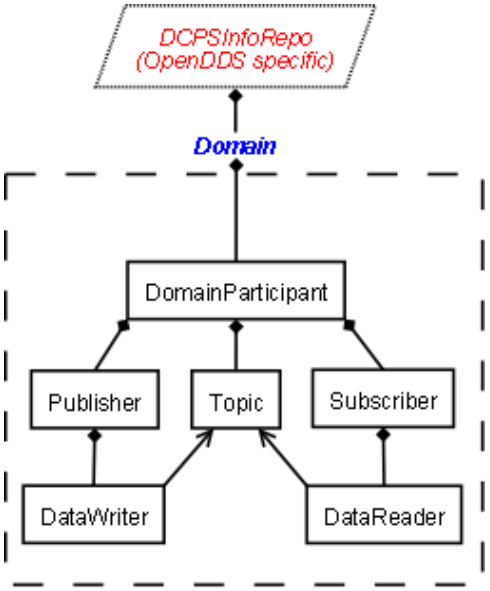

Figure 2.7: DDS components and the OpenDDS specific component [14].

2.4 Operating Conditions

The robotic systems are teleoperated over a wireless network. The latency of a wireless network is affected by the *signal-to-noise ratio (SNR)* and *data size*. The SNR does not directly influence the latency. However, it does indirectly influence the latency via network overhead (simulates the number of clients on a network) and packet loss. The data size determines how many data packets are required to transmit the data. The more data packets it requires the higher the latency will be, since the packets are consecutively transmitted and each packet has a certain latency. During the tests in this study, latency was also found to be affected by DDS QoS settings (discussed in Section 2.3.1), OSs, resource-rich and resource-restricted hardware.

The below mentioned operating conditions influence the latency and are therefore manipulated in this research to measure whether acceptable or preferred latency levels can be acquired. These conditions are described consequently in the remainder of this chapter:

- data size,
- signal power,
  - packet loss, and
  - network overhead.

2.4.1 Data Size

Data size indicates the number of bytes a certain data sample consists of. Before data is sent over a network, they are divided into packets. Each packet can have a maximum number of bytes. This is dependent on the Maximum Transmission Unit (MTU). If the data size is larger than the MTU, it will be divided into multiple packets. For example, if the data size is 2,000 bytes and the MTU is 1,500 bytes, the data will be divided into two packets as shown in Figure 2.8.

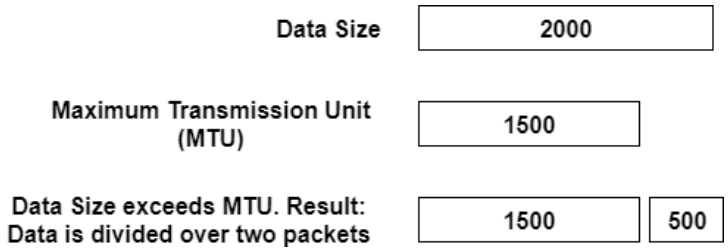

Figure 2.8: Data being divided into multiple packets when it exceeds the MTU.

Data, such as movement commands and video images sent to the robotic system and controller respectively, have data sizes ranging from a few bytes to multiple kilobytes. Therefore, the data sizes used in this study ranged from 4 bytes to 16 kilobytes and increased by 2^i bytes, with i starting at 2, and increasing by 2 up to 14. This data size range was also used by a previous study of J. Yang [18] mentioned in the introduction.

2.4.2 Signal Power

Signal power [9] divided by the amount of noise power (often measured in decibels (dBm)) produces the SNR. The SNR is a ratio that indicates the efficiency of a signal with respect to noise. The higher this ratio is, the higher the probability that all transmitted data will be received correctly, and vice versa. This ratio is influenced by interference from other senders or objects, such as walls or trees absorbing and decreasing the signal's power. A good SNR (>1) means that the signal power is stronger than the noise. This results in a lower bit error ratio (BER). A bad SNR (<1) means that the BER will be higher. The BER [4] is defined as "the number of incorrect bits divided by the total number of transfered bits", often expressed as a percentage. At a certain BER the receiving side cannot reconstruct the original packet, which means that the packet is lost. The terminology for this is packet loss, expressed as packets lost per total packets sent.

The BER also influences the modulation technique [11] used inside a wireless network. Modulation is a technique that varies the carrier signal's amplitude, frequency, or phase with a signal that contains the information to be transmitted (stream of bits). A more complex modulation technique results in a higher bandwidth [2], but requires a high SNR, and vice versa.

The SNR or BER indirectly influences latency via network overhead [12]. As stated earlier, the SNR or BER influences the complexity of the modulation technique used to encrypt and decrypt data, which affects the maximum bandwidth. A decrease in SNR can lead to the system using a less complex modulation technique, which decreases the maximum bandwidth. During the tests it was found that the latency increased when the required bandwidth exceeded the maximum bandwidth. For example, the required bandwidth might be 90 Mbit/s, 80 Mbit/s of which is network overhead. The maximum bandwidth is 100 Mbit/s. At some point the SNR decreases, leading to a less complex modulation technique, which results in a new maximum available bandwidth of 80 Mbit/s. This leads to an increased latency, because the required bandwidth of 90 Mbit/s exceeds the maximum bandwidth of 80 Mbit/s. If there was no network overhead, or if the SNR did not cause the maximum bandwidth to lower, the latency would not have been affected. Therefore, the SNR indirectly influences latency via network overhead.

2.5 Conclusion

In order to reliably teleoperate robotic systems the low-level mission has to be accomplished in a satisfactory manner under specified operating conditions. The low-level mission is to receive data from a robotic system and controller within a specified time period. This time period has to be equal or lower than the latency requirements stated in table 2.2 in order to be satisfactory. The latency requirements are different for each task. A task can either be impressive or less impressive. Impressive tasks are compared to fast-paced video games and less impressive tasks are compared to slow-paced video games. The specified operating conditions are packet loss, network overhead, OS and hardware, using DDS QoS Reliability 'best effort' and 'reliable', and data sizes ranging from 4 to 16,384 bytes.

	Latency (ms)	
Task	Acceptable	Preferred
Less impressive	200	50
Impressive	75	25

Table 2.2: The acceptable and preferred latency requirements (in milliseconds) to reliably teleoperate robotic systems used by the NPN for impressive and less impressive tasks.

This project uses OpenDDS to validate whether DDS can be used to reliably teleoperate robotic systems used by the NPN. OpenDDS is an open source C++ implementation of DDS and used by Project Robots voor Veiligheid (Robots for Security, RoVe), since it is free and has all the components of DDS.

Chapter 3

Method

The method that was used to investigate if Data Distribution Service (DDS) can be used to reliably teleoperate robotic systems used by the National Police of the Netherlands (NPN) is addressed in this chapter. First the test setup used to validate this is discussed in Chapter 3.1. This section also addresses how latency is measured and how the operating conditions of packet loss, network overhead, Operating System (OS) and hardware are introduced to the tests. The test setup uses a wired network in order to obtain reliable and repeatable results, since a wireless network is more sensitive to noise. However, the results must be applicable to a wireless network, and therefore require a conversion. This conversion is addressed in section 3.2.

3.1 Test Setup

The test setup shown in Figure 3.1, which was used to validate whether DDS complies with the latency requirements, consists of four components: a *master*, a *slave*, a *network overhead*, and a router. For the master and network overhead a resource-rich laptop was used, and for the slave either a resource-rich laptop or a resource-restricted Raspberry Pi 3 was used. They were all connected to the same router via an Ethernet cable (CAT 5). During the tests, the master and the slave were always connected to the router. The network overhead laptop was only connected when the tests required network overhead.

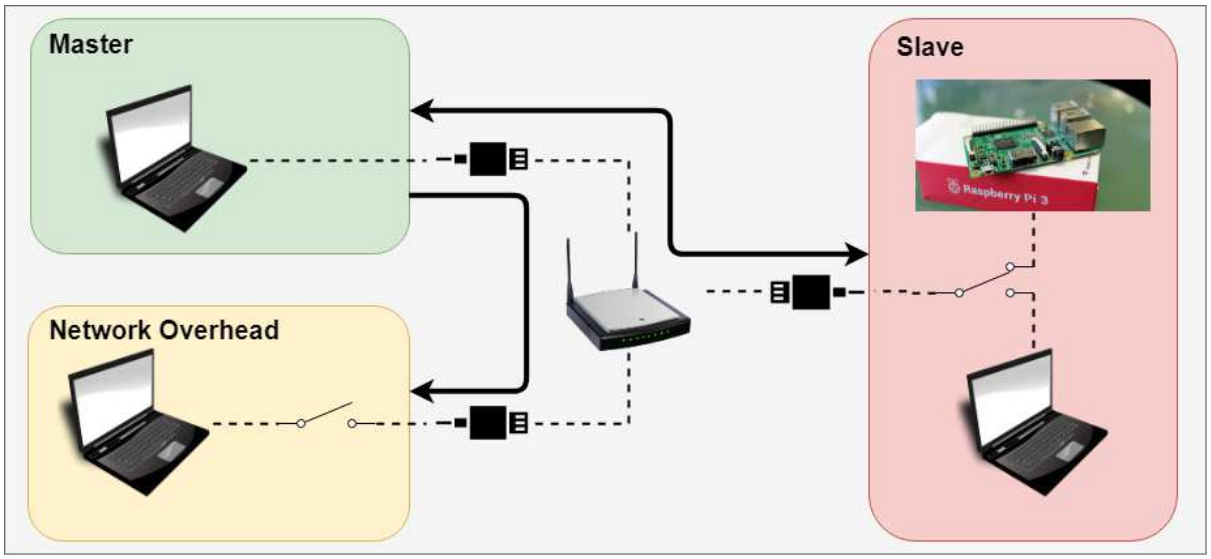

Figure 3.1: Global overview of the test setup.

Two separate scripts were running on the master: a Publisher and a Subscriber. The Publisher sends data of a specific size to the slave and the Subscriber receives data from the slave. The data is sent at a fixed interval to simulate a stable frequency of sending data. The reason that the Publisher and the Subscriber were separate was to prevent incoming data from getting timed later due to overhead. Both receive times and send times were stored inside a text file.

The slave had one script running that combined the Publisher and the Subscriber. The primary task of the slave was to resend the data received by the master. The reason that they were combined was to avoid thread switches and synchronization issues of shared variables, making the study less error prone and complex. However, a disadvantage is that in certain situations data could accumulate over time, resulting in an increased latency. This only occurs when the execution time is greater than the send interval of the master's Publisher. The execution time is defined as the time between receiving data from the master and sending this data back to the master. To avoid that data accumulates over time, the frequency was set to 50 Hertz (Hz), which is an interval of 200 ms and larger than the execution time of $\simeq 0.3$ ms.

3.1.1 Latency Measurement

The robotic system sends video images to the controller, and the controller sends control signals to the robotic system. These two processes run parallel and can be interpreted as two separate one-way communication channels. This means that the measured latency should also be measured as one-way. Since it is difficult to measure one-way latency, it is often measured as two-way, or round-trip time (RTT).

Measuring one-way latency requires a test setup with one of the following options:

1. a synchronized timer of sender and receiver;
2. a third party timer, such as the Internet, a router or another device to retrieve the time; or
3. measuring the RTT and then dividing it by two to get the one-way latency.

The first option is accurate, but requires adjusting both internal timers and initiating them at the exact same time. This method is difficult to accomplish, and therefore was not chosen for this study. The second option is not reliable because the inaccuracy of Internet time is in the order of milliseconds, and retrieving the time over a network is not consistent due to jitter. The third option measures the average latency of two samples, since the data is sent and received twice. This does not affect the average latency, but it does affect the accuracy of the one-way latency measurement when the two one-way latencies are not equal. Assuming that the error inherent in the inaccuracy of option three is less than the error inherent in option two, the third option was chosen for this study: measuring the RTT and then dividing it by two.

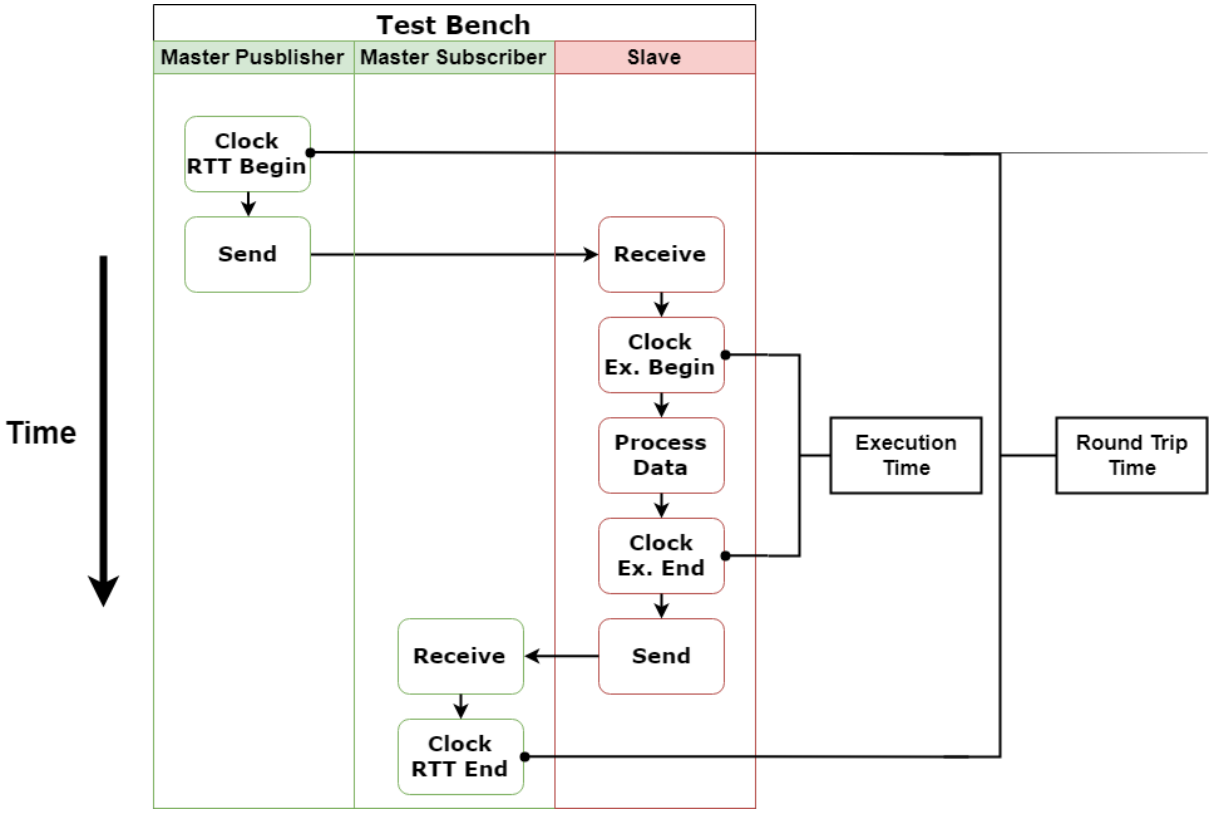

Figure 3.2: A sequence diagram of the test setup.

For this test setup, the latency was measured by clocking the data's travel time from master to slave and back to master, shown in Figure 3.2. The process starts with the master Publisher clocking (Clock RTT Begin) the time right before the data is transmitted to the slave, starting the measurement to compute the RTT latency. After the slave receives the data, the time is clocked (Clock Ex. Begin) to measure the execution time. The time is clocked again before the slave sends data to the master Subscriber (Clock Ex. End), stopping the execution time measurement. When the master Subscriber receives the data, the time is clocked one last time (Clock RTT End), stopping the RTT latency measurement.

RTT latency and execution time measurements were done for each data sample sent. There was a total of 10,000 data samples sent, each with their own latency. The RTT and execution time were used to compute the required one-way latency as shown in Equation 3.1.

$$Latency = \frac{Round\_Trip\_Time - Execution\_Time}{2} \tag{3.1}$$

3.1.2 Packet Loss

This section discusses the method used for the first research sub-question of whether the DDS meets the latency requirement when subjected to different data sizes and different types of DDS Quality of Service (QoS) settings for the operating condition packet loss.

The operating condition of packet loss was validated by sending data between the master and the slave (Figure 3.1) while the program qdisc induced packet loss. The amounts of packet loss induced were 0%, 8%, and 16%, because quality is affected at around 10% [8]. Each of these three percentages of packet loss were individually tested against seven different sizes of data (addressed in Section 2.4), and DDS QoS Reliability 'reliable' and 'best effort'.

The percent of samples received that was expected under the DDS QoS Reliability setting 'best effort' can be computed using Equation 3.2.

$$Data\_Received = \left(\frac{100 - packet\_loss}{100}\right)^{\left\lceil \frac{data\_size}{MTU} \right\rceil} \tag{3.2}$$

For example, with 8% packet loss, a data size of 4,096 bytes, DDS QoS Reliability 'best effort', a Maximum Transmission Unit (MTU) of 1,500 bytes results in 78% of samples received (see Equation 3.4).

$$Data\_Received = \left(\frac{100 - 8}{100}\right)^{\left\lceil \frac{4096}{1500} \right\rceil} \tag{3.3}$$

$$= 0.92^3 = 0.78 = 78\% \tag{3.4}$$

The DDS QoS Reliability 'reliable' should receive all transmitted samples.

As mentioned earlier, packet loss was created with the program qdisc. This program has three important commands that were used during the tests:

- adding packet loss: 'sudo tc qdisc add dev NN root netem loss PP%',
- changing packet loss: 'sudo tc qdisc change dev NN root netem loss PP%', and
- removing packet loss: 'sudo tc qdisc del dev NN root'.

The 'NN' stands for the network name to induce the packet loss onto, and 'PP' for the amount of packet loss as a percent.

3.1.3 Network Overhead

The method used for the second research sub-question of whether DDS meet latency requirements when subjected to different data sizes and different types of DDS QoS settings for the operating condition of network overhead is discussed in this section.

Sending additional "useless" data (besides the original data) between the master and the network overhead laptop creates network overhead, as shown in Figure 3.1. It was not possible to generate network overhead using two network overhead laptops because the maximum bandwidth of the laptops that were used was less than the maximum bandwidth of the router. The router that was used was a Linksys WRT 1200 AC with a 1,000 megabit bandwidth, whereas the laptops had a 100 megabit bandwidth each. This means that a maximum of 10% bandwidth could be occupied. However, when sending data from the master to the network overhead, 100% of the bandwidth could be occupied. The downside of this method is that it generates overhead at the master. The latter method was chosen for this research because the study needed to use 100% of the bandwidth in order to see results such as increased latency and packet loss.

The operating condition of network overhead was validated by transferring data between the master and the slave while the program iPerf induced network overhead in the background. The amounts of network overhead that were induced to the tests were 0%, 50%, and 95% when the DDS QoS Reliability 'reliable' was used, and 0%, 50%, 90%-99% with a step size of 1% when the DDS QoS Reliability 'best effort' was used. During the first tests (0%, 50%, and 95%), network overhead had the same effect on latency for both DDS QoS Reliability settings. 'Best effort' performed better (see Appendix A.3), so this study chose to test the DDS QoS Reliability 'best effort' more thoroughly. The reason for a 1% step size between 90% and 99% network overhead was because the effects of network overhead were observable at around this percentage during the tests. Each percentage of network overhead was tested against seven different data sizes (addressed in Section 2.4).

The program iPerf was used to create network overhead. This program is used to send data with a specific data size to another device. It can also be used to measure active bandwidth, packet loss, and jitter. For this study, it was only used to induce network overhead and measure maximum bandwidth. The following command was run on the master:

iperf -c IP_NO -u -t DURATION -b BANDWIDTH

'IP_NO' was the IP address of the network overhead laptop. 'DURATION' was set to a value larger then the expected test duration, and 'BANDWITH' was the bandwidth to be occupied in megabits per second (Mbit/s). The '-c' stands for client and '-u' stands for User Datagram Protocol (UDP). The value of 'BANDWIDTH' was computed by first computing the maximum bandwidth (also using this program), and then dividing it by 100 and multiplying it with the network overhead percentage. The following command was ran on the network overhead laptop:

iperf -s -u

Here, '-s' stands for server and '-u' for UDP.

3.1.4 Operating System and Hardware

This section discusses the method used to investigate the third research sub-question of whether DDS meets the latency requirements when subjected to different data sizes and different types of DDS QoS settings for the operating condition of OS and hardware.

Validation of the latency requirement did not require the utilities of robotic systems and controllers—merely the hardware and software. Laptops and single board computers were used. This hardware and software is similar to the robotic systems and controllers, but the advantage is that this hardware and software is more flexible, accessible, and convenient to use during tests. Therefore, this study chose to use laptops and single board computers instead of robotic systems and controllers.

There are multiple types of robotic systems and controllers. This controller has hardware similar to a tablet or laptop, whereas the robotic system has hardware similar to a Raspberry Pi 3.

The operating condition of OS and hardware was validated by transferring data between the master and the slave (Figure 3.1) using different OSs and hardware. The OSs included Windows 10, Ubuntu 16.04 and Raspbian, and the hardware consisted of a laptop (resource-rich) and a Raspberry Pi 3 (resource-restricted). The OS Raspbian was only used on the Raspberry Pi 3, and the OSs Windows 10 and Ubuntu 16.04 were only used on a laptop. The reason that these OSs and hardware were chosen was because the controller can use Windows 10 or Ubuntu as an OS and the robotic system can use Ubuntu or Raspbian as an OS. The OSs and hardware were tested as follows, with the OS on the left being used on the master and the OS on the right being used on the slave:

- Ubuntu 16.04 (Laptop) ⟷ Ubuntu 16.04 (Laptop)
- Ubuntu 16.04 (Laptop) ⟷ Raspbian (Raspberry Pi 3)
- Windows 10 (Laptop) ⟷ Ubuntu 16.04 (Laptop)
- Windows 10 (Laptop) ⟷ Raspbian (Raspberry Pi 3)

The following hardware specs were used during the tests:

- Windows 10 on Master[1]:
  - processor: Intel(R) Core(TM) i5-2410M CPU @ 2.30 GHz
  - RAM: 4GB
  - system type: 64-bit
- Ubuntu 16.04 LTS on Master[1]:
  - processor: Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz
  - RAM: 4GB
  - system type: 64-bit OS
- Ubuntu 16.04 LTS on Slave:
  - processor: Intel(R) Core(TM) 2 DUO CPU U9300 @ 1.20GHz * 2
  - RAM: 1.8 GB
  - OS type: 64-bit
- Raspbian GNU/Linux 8 (jessie) on Slave:
  - SoC: Broadcom BCM2837
  - CPU: 4* ARM Cortex-A53 @ 1.2GHz
  - RAM: 1GB LPDDR2 (900 MHz)

Each of these four setups was tested using seven different sizes of data (addressed in Section 2.4), and the DDS QoS Reliability setting 'best effort'.

3.2 Conversion

The robotic systems are usually teleoperated over a wireless network. However, a wired network was chosen for testing because a wired network is more consistent and less sensitive to noise than a wireless network. The signal-to-noise ratio (SNR) for a wired network is constant and low, meaning that the results produced during the tests were affected only by the user's input. An anechoic chamber, which is used to reduce noise, was not available for this research.

[1] duoboot

The consistency provided by the wired network is noticeable in the latency spread, or standard deviation, between the results of tests using a wireless network (Figure 3.3) and the results of tests using a wired network (Figure 3.4). The standard deviation relative to the median for a wired network is between 2% and 10%, whereas the standard deviation for a wireless network is between 80% and 210%, as seen in Table 3.1. Both tests, called baseline tests, were performed under the same operating conditions with a distance of 20 cm between the master and the slave. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OS Ubuntu 16.04 ⟷ Raspbian.

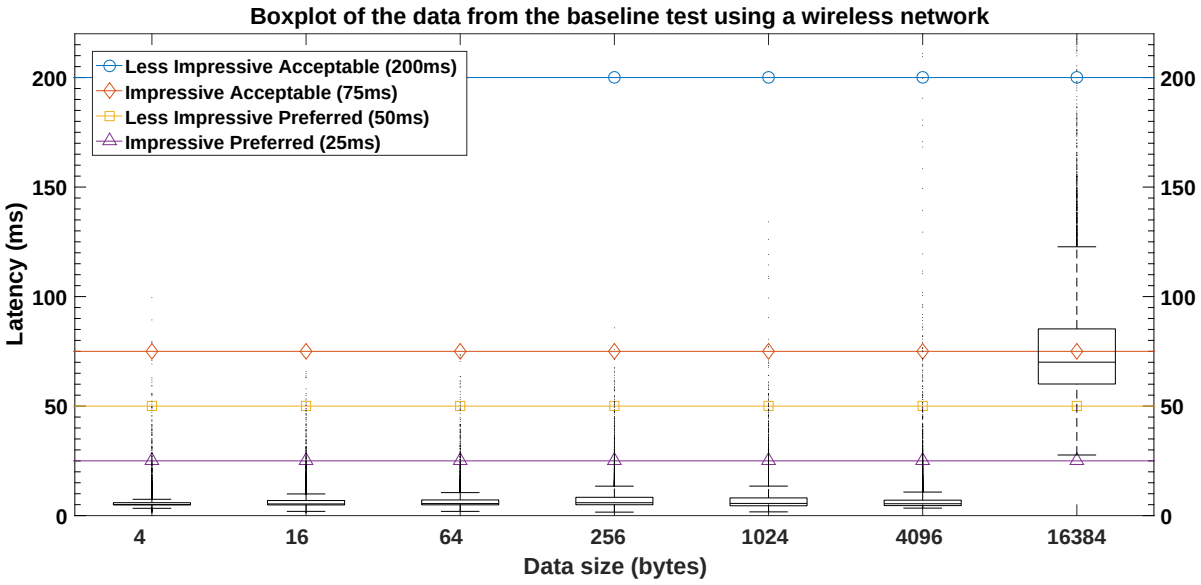

Figure 3.3: The latency boxplot of the baseline test over a wireless network with the latency requirements shown as different colours and symbols. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

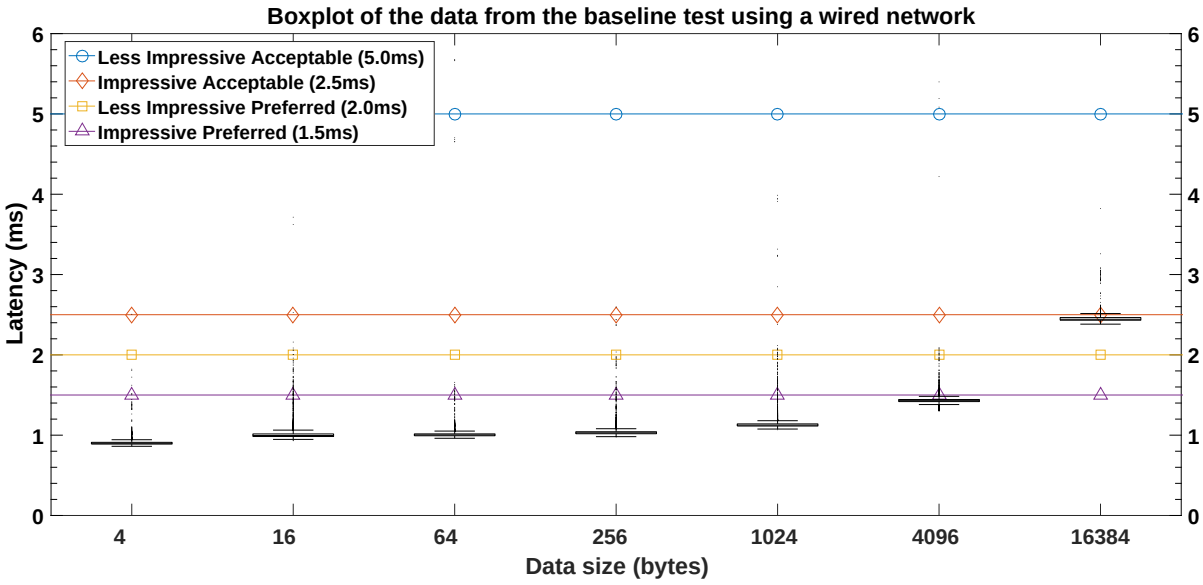

Figure 3.4: The latency boxplot of the baseline test over a wired network together, with the latency requirements shown as different colours and symbols. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

The validation of the test results regarding the latency requirements using a wired network could not be performed directly. (Figure 3.3) had a greater average latency compared to the baseline test using a wired network (Figure 3.4). The baseline test using a wireless network also had a higher standard deviation, shown in Table 3.1. Because the latencies and standard deviations between the two network types were unequal, the test results of introducing packet loss, network overhead, and different OSs performed over

Standard deviation	Data size (bytes)						
	4	16	64	256	1024	4096	16384
Wired network	6.5	12.6	11.6	7.0	9.5	6.0	1.9
Wireless network	85.6	82.6	85.1	94.3	125.0	210.8	39.0

Table 3.1: This table shows the standard deviation relative to the median expressed in percentages for different data sizes and networks.

a wired network need to be scaled accordingly to reflect the conditions of the wireless network used by the NPN.

One option was to scale the latency of each measured data sample by a certain amount. This would require adding a certain offset and standard deviation to the measured data samples. Another option was to scale the latency requirements. Both options introduce errors in the probability of meeting the latency requirements. The error rate of scaling each measured data sample is influenced by the offset and standard deviation. The error rate of scaling the latency requirements is influenced by the scale value. Scaling the latency requirement has fewer variables that influence the error rate than scaling each measured data sample. Therefore, this study chose to scale the latency requirements.

The probabilities of meeting the **less impressive acceptable (200 ms)**, **impressive acceptable (75 ms)**, **less impressive preferred (50 ms)**, and **impressive preferred (25 ms)** latency requirements for the baseline test using a wireless network (shown in Table 3.2) were used as a guideline to scale the latency requirements for the baseline test using a wired network. As seen in Figure 3.3, for the baseline test using a wireless network, the **impressive acceptable (75 ms)** latency requirement is nearly equal to the latency's median of 16,384 bytes data size. The corresponding probability of meeting this latency requirement is 60.8%. The other data sizes have a probability of about 100% of meeting this latency requirement.

Network	Latency requirement	Number of packets (data size in bytes)		
		1 (4, 16, 64, 256 and 1024)	3 (4096)	11 (16384)
Wireless	Less impressive acceptable	100.0	100.0	99.5
	Impressive acceptable	100.0	99.6	60.8
	Less impressive preferred	99.8	99.1	6.1
	Impressive preferred	98.5	97.0	0.0
Wired	Less impressive acceptable	100.0	100.0	100.0
	Impressive acceptable	100.0	100.0	61.2
	Less impressive preferred	99.9	99.9	0.0
	Impressive preferred	99.4	97.0	0.0

Table 3.2: Probability of a wired and wireless network meeting the latency requirements.

The latency's median at a data size of 16,384 bytes for the wired network baseline test, shown in Figure 3.4, is around 2.4 ms. Scaling the **impressive acceptable** latency requirement slightly higher to 2.5 ms for the baseline test using a wired network resulted in a probability of 61%. This is relatively close to the 60.8% mentioned earlier for the baseline test using a wireless network. The other data sizes have a probability of 100% of meeting the **impressive acceptable (2.5 ms)** latency requirement.

The latency difference between the **impressive acceptable** latency requirement of a wireless network (75 ms) and a wired network (2.5 ms) was used to scale the other latency requirements. The scale value was determined as follows:

$$\frac{75.0\ ms}{2.5\ ms} \simeq 30.0 \tag{3.5}$$

As determined by the equation, the new **less impressive acceptable (200 ms)** latency requirement was downscaled to 6.7 ms.

$$\frac{200.0\ ms}{30.0} \simeq 6.7\ ms \tag{3.6}$$

The **less impressive preferred (50 ms)** latency requirement was downscaled to 1.7 ms.

$$\frac{50.0\ ms}{30.0} \simeq 1.7\ ms \tag{3.7}$$

The **impressive preferred (25 ms)** latency requirement was downscaled to 0.8 ms:

$$\frac{25.0\ ms}{30.0} \simeq 0.8\ ms \tag{3.8}$$

The new latency requirements, along with the baseline test results using a wired network, are shown in Figure 3.5. The **impressive preferred (0.8 ms)** latency requirement is less than the measured latencies, with a corresponding probability of 0% for all data sizes. For the baseline test using a wireless network, this was about 100%, except for the 16,384 data size, which was also 0%. Therefore, for this study it was decided to increase the **impressive preferred (0.8 ms)** latency requirement to 1.5 ms and the **less impressive preferred (1.7 ms)** latency requirement to 2.0 ms, keeping the **impressive acceptable (2.5 ms)** latency requirement at 2.5 ms and decreasing the **less impressive acceptable (6.7 ms)** latency requirement to 5.0 ms, as shown in Figure 3.4. These values are chosen so that the original latency requirement ratio from the wireless baseline test stayed intact. The original latency requirement ratio was 0.5 ms for every 25 ms, starting at 1.5 ms.

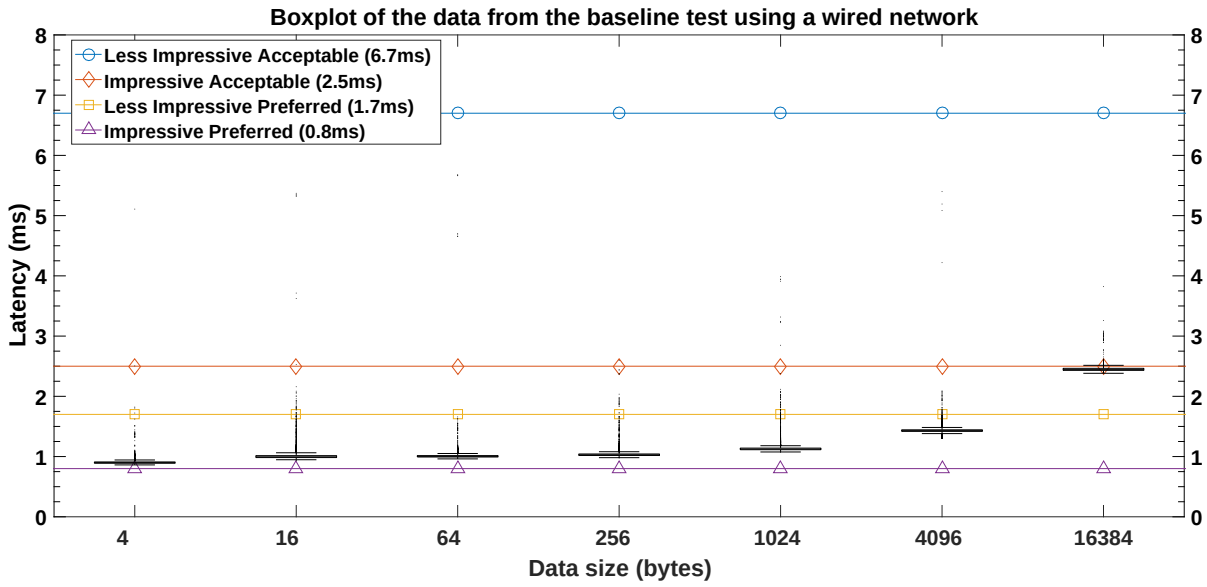

Figure 3.5: The latency boxplot of the baseline test over a wired network with the latency requirements shown as different colours and symbols. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

The resulting probabilities of meeting the **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)** latency requirements for the baseline test using a wired network was almost equal to the probabilities of meeting the corresponding latency requirements for the baseline test using a wireless network, as shown in Table 3.2. The error here is relatively small ($\approx 1\%$).

In conclusion, the results were validated using the latency requirements shown in Table 3.3.

	Latency (ms)	
Task	acceptable	preferred
Less impressive	5.0	2.0
Impressive	2.5	1.5

Table 3.3: The acceptable and preferred latency requirements (in milliseconds) for impressive and less impressive tasks used in this study.

For more detailed results, such as median and maximum latency for both baseline tests, see Table A.1 in Appendix A.

Chapter 4

Results

This chapter addresses the results of the method used to validate whether Data Distribution Service (DDS) can be used to reliably teleoperate robotic systems used by the National Police of the Netherlands (NPN) under specified operating conditions. These operating conditions include packet loss, network overhead, and Operating System (OS) and hardware, as addressed in Sections 4.1, 4.2, and 4.3 respectively.

The information presented in these sections will cover a global interpretation of the results and the probability of meeting the **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)** latency requirements. The discussion and conclusions regarding the implications of these results in determining whether DDS can be used to reliably teleoperate robotic systems will be covered in Chapter 5.

4.1 Research Sub-Question I: Packet Loss

This section presents results regarding the research sub-question of whether DDS meets latency requirements when subjected to different data sizes (ranging from 4 to 16,384 bytes) and DDS Quality of Service (QoS) Reliability 'reliable' and 'best effort' under the operating condition of packet loss. The operating condition of packet loss has been validated by sending data between the master and the slave while the program qdisc simulated a certain percentage of packet loss.

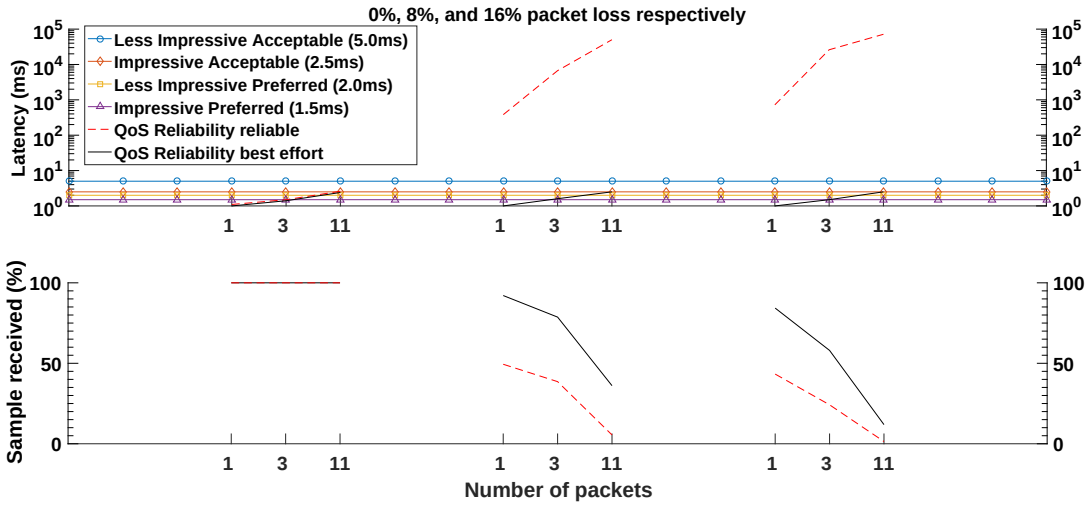

Figure 4.1: The latency boxplot of 0%, 8%, and 16% packet loss for DDS QoS Reliability 'reliable' (striped) and DDS QoS Reliability 'best effort' (line) from left to right is shown in the upper graph along with the four latency requirements. The lower graph shows the corresponding percentage of samples received. Besides these packet loss percentages, the following operating conditions were used: 0% network overhead, DDS QoS Reliability 'reliable' and 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

The results of introducing different percentages of packet loss and varying data sizes show that when using DDS QoS Reliability 'reliable' the overall latency is higher than when using DDS QoS Reliability 'best effort' (Figure 4.1). The results also show that DDS QoS Reliability 'best effort' has a higher data reception rate than DDS QoS Reliability 'reliable'.

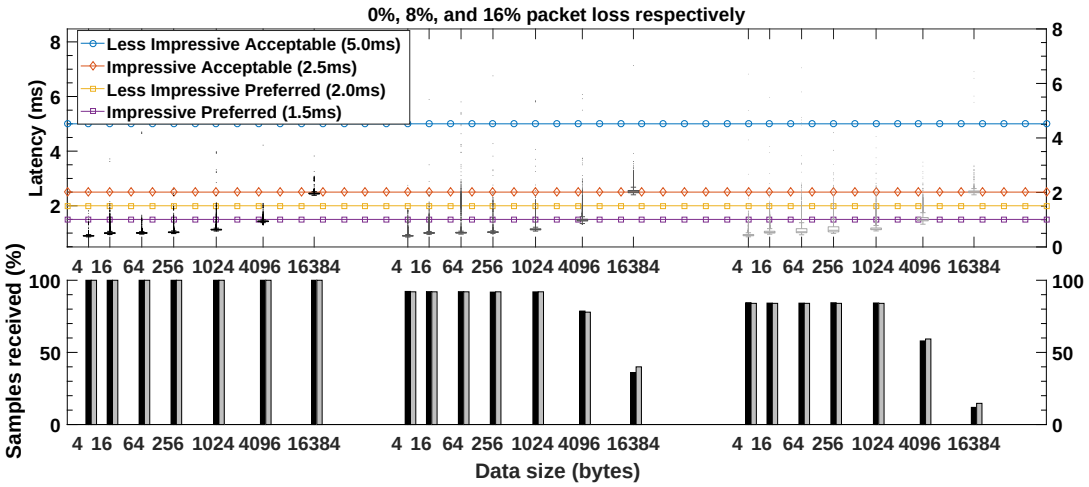

Figure 4.2: A latency boxplot of 0%, 8%, and 16% packet loss from left to right is shown in the upper graph along with the four latency requirements. The lower graph shows the corresponding percentage of samples received for the test results (black) and the percentage that was expected to be received (grey). Besides these packet loss percentages, the following operating conditions were used: 0% network overhead, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

For DDS QoS Reliability 'best effort', the average latency remained the same for different percentages of packet loss, as shown in Figure 4.2. It should be noted that lost data, which has an infinite latency, was not included in the results to measure the average latency. This would make the average latency infinite, and therefore unusable. The lost data was used to determine the data received percentage shown in Figure 4.2, in the lower bar graph. The expected amounts of data received for DDS QoS Reliability 'best effort' are shown in Table 4.1, and are almost equal to the test results presented in Figure 4.2.

Packet loss (%)	Number of packets (data size)		
	1 (4, 16, 64, 256 and 1,024 bytes)	3 (4,096 bytes)	11 (16,384 bytes)
0	100.0	100.0	100.0
8	92.0	77.9	40.0
16	84.0	59.3	14.7

Table 4.1: The expected percentage of data received is shown in the white area. The amount of packet loss applied to the tests is on the vertical axes (light grey), and the number of packets or data size is on the horizontal axes (dark grey). Packet value one is an average of five data sizes, because the maximum amount of bytes one packet can contain is 1,500 bytes and the given five data sizes are all below this maximum.

Using the DDS QoS Reliability 'reliable' and introducing packet loss of 8% and higher has a 4.9% or lower probability (sample receive rate (%) * latency requirement (%)) for all data sizes of meeting the four latency requirements (5.0 ms, 2.5 ms, 2.0 ms, and 1.5 ms).

When using DDS QoS Reliability 'best effort' and inducing 8% packet loss for data sizes up to 1,024 bytes, the probability to meet the **less impressive acceptable (5.0 ms)** requirement has the highest probability (92.1%). At greater data sizes, packet loss percentages or the other latency requirements the probability is lower ranging from 92.1% to 0%.

For detailed results, see Appendix A, Table A.2. This table presents the median, standard deviation, worst case, samples received, and probability of meeting the latency requirements for DDS QoS Reliability settings 'reliable' and 'best effort'; 0%, 8%, and 16% packet loss; and for one (equivalent to 4, 16, 64, 256, 1,024 bytes data size), three (equivalent to 4,096 bytes data size), and 11 (equivalent to 16,384 bytes data size) packets.

4.2 Research Sub-Question II: Network Overhead

This section presents the results regarding the research sub-question of whether DDS meets latency requirements when subjected to different data sizes (ranging from 4 to 16,384 bytes) and DDS QoS Reliability settings 'reliable' and 'best effort' used under the operating condition of network overhead. The operating condition network overhead were validated by sending data between the master and the slave laptops while the program iPerf induced a certain amount of network overhead.

The results of inducing different percentages of network overhead and sending varying data sizes show that the latency and percentage data received remained the same when the used bandwidth was less than the available bandwidth. However, when the used bandwidth exceeded the available or maximum bandwidth, the latency and packet loss increased. The required bandwidth to exceed the available or maximum bandwidth can be computed using Equation 4.1. The required bandwidth in order to exceed the maximum bandwidth used for this test setup is presented in Table 4.2. For this test setup, the bandwidths required in order to exceed the maximum bandwidth for package sizes one, three, and 11 were 99.6%, 98.3%, and 93.2% respectively. Comparing the test results shown in Figures 4.3 and 4.4 reveals that at the computed required bandwidth percentages (93.2%, 98.3%, and 99.6%, with a $\approx 1\%$ margin) the latency increases and packets received decreases. When the used bandwidth was equal to or higher than the maximum bandwidth, the latency increased and the sample receive rate decreased.

$$Required\_Bandwidth = Maximum\_Bandwidth - DDS\_Bandwidth \tag{4.1}$$

	Number of packets (data size)		
	1 (4, 16, 64, 256 and 1,024 bytes)	3 (4,096 bytes)	11 (16,384 bytes)
Required Network Overhead (%)	99.6	98.3	93.2

Table 4.2: The required bandwidth in order to exceed the maximum bandwidth is shown in the white area. These values were computed by subtracting the used bandwidth from the maximum bandwidth. The maximum bandwidth was 96 Megabits/s during the tests.

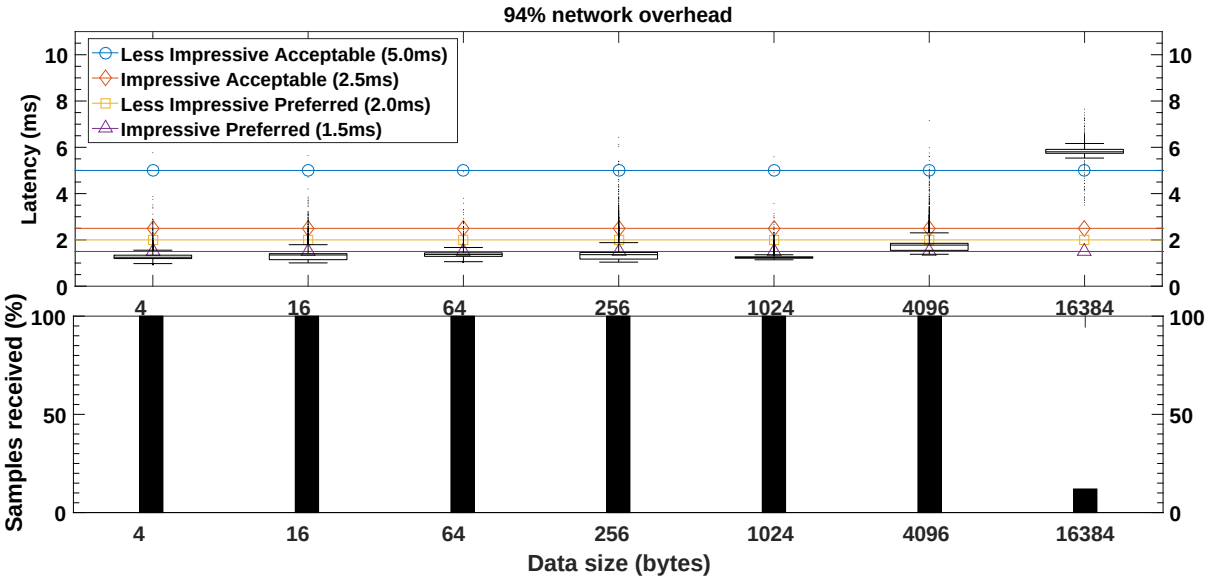

Figure 4.3: The latency boxplot of 94% network overhead is shown in the upper graph along with the four latency requirements. The lower graph shows the corresponding percentage of samples received. Besides the 94% network overhead, the following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

The overall probability of meeting the latency requirements remained equal to the baseline tests except when the total bandwidth exceeded the maximum bandwidth. The probability of meeting the **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and

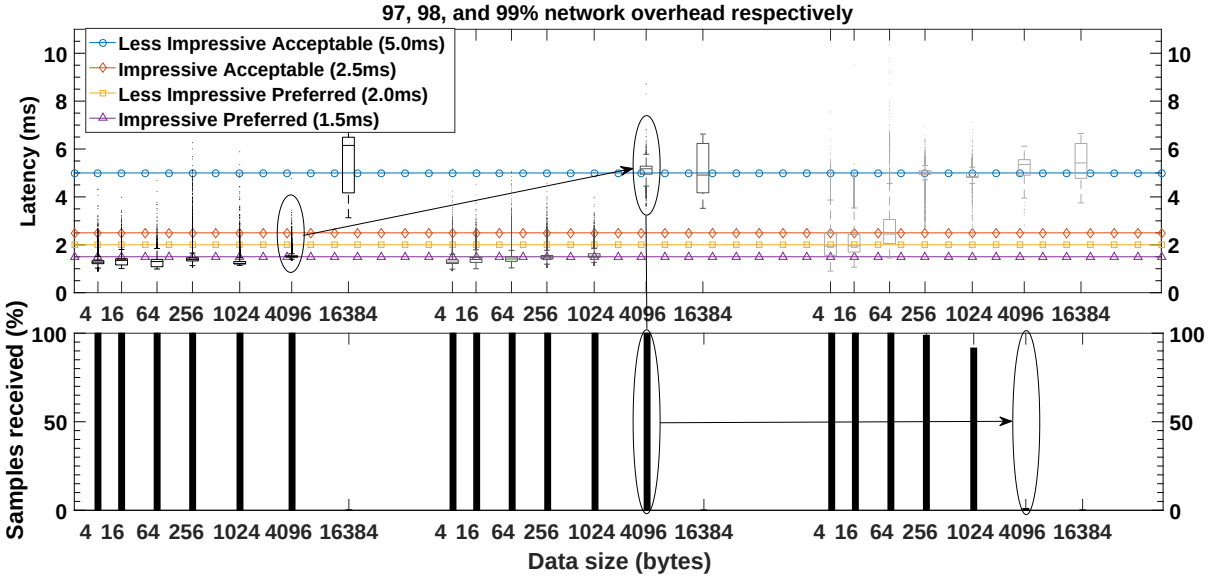

Figure 4.4: The latency boxplot of 97%, 98%, and 99% network overhead from left to right is shown in the upper graph along with the four latency requirements. The lower graph shows the corresponding percentage of samples received. Besides these network overhead percentages, the following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

impressive preferred (1.5 ms) latency requirements at the computed required bandwidth percentages (93.2%, 98.3%, and 99.6%) or higher were 0.0%.

For detailed results, see Appendix A, Table A.3. This table presents the median, standard deviation, worst case, samples received, and probability of meeting the latency requirements under DDS QoS Reliability 'reliable' and 'best effort'; between 0% and 99% network overhead; and for one (equivalent to 4, 16, 64, 256, or 1,024 bytes data size), three (equivalent to 4,096 bytes data size), and 11 (equivalent to 16,384 bytes data size) packets.

4.3 Research Sub-Question III: Operating Systems and Hardware

This section presents the results regarding the research sub-question of whether DDS meets the latency requirement when subjected to different data sizes (ranging from 4 to 16,384 bytes) and DDS QoS Reliability 'best effort' used with different OS and hardware. The OS and hardware conditions were validated by sending data between the master and the slave while using the following OSs and hardware. The OS on the left is used on the master, and the OS on the right is used on the slave:

- Ubuntu 16.04 (Laptop) ⟷ Ubuntu 16.04 (Laptop)
- Ubuntu 16.04 (Laptop) ⟷ Raspbian (Raspberry Pi 3)
- Windows 10 (Laptop) ⟷ Ubuntu 16.04 (Laptop)
- Windows 10 (Laptop) ⟷ Raspbian (Raspberry Pi 3)

The results shown in Figure 4.5 show that the OS influences the latency. The OS Ubuntu 16.04 had the lowest average latency, followed by Raspbian and then Windows 10.

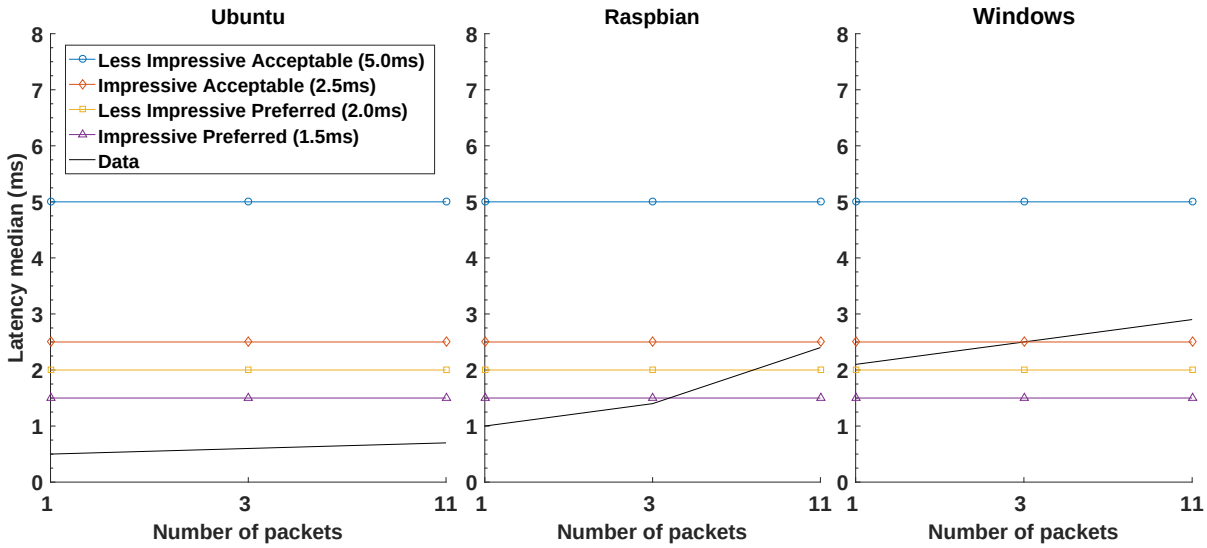

Figure 4.5: The median latency of OSs Ubuntu 16.04 ⟷ Ubuntu 16.04 is shown on the left, OSs Ubuntu 16.04 ⟷ Raspbian in the middle, and OSs Windows 10 ⟷ Ubuntu 16.04 on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

The results of using resource-rich hardware versus resource-restricted hardware, shown in Figures 4.6 and 4.7, indicate that when using resource-rich hardware the latency median was lower than when using resource-restricted hardware. The latency median of transferring data between two resource-rich hardware (shown on the left) was lower than the transfer of data between a resource-rich and resource-restricted hardware (shown on the right). For this test setup, resource-rich hardware refers to the laptops running Windows 10 and Ubuntu 16.04, and resource-restricted hardware to the Raspberry Pi 3 running Raspbian.

The overall probability for all OSs and hardware of meeting the **less impressive acceptable (5.0 ms)** latency requirement was above 99.3%, except when sending data between a resource-rich hardware with Windows 10 and a resource-restricted hardware with OS Raspbian. For sending data with a size of 16,384 bytes, the probability was 0%.

The overall probability of meeting the **less impressive acceptable**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)** latency requirements was above 97.0% when sending data between a resource-rich hardware with OS Ubuntu 16.04 and a resource-restricted hardware with OS Raspbian or another resource-rich hardware with OS Ubuntu 16.04. The only exception was when sending data of 16,384 bytes between a resource-rich hardware with OS Ubuntu 16.04 and a resource-restricted hardware with OS Raspbian. The probability of meeting the

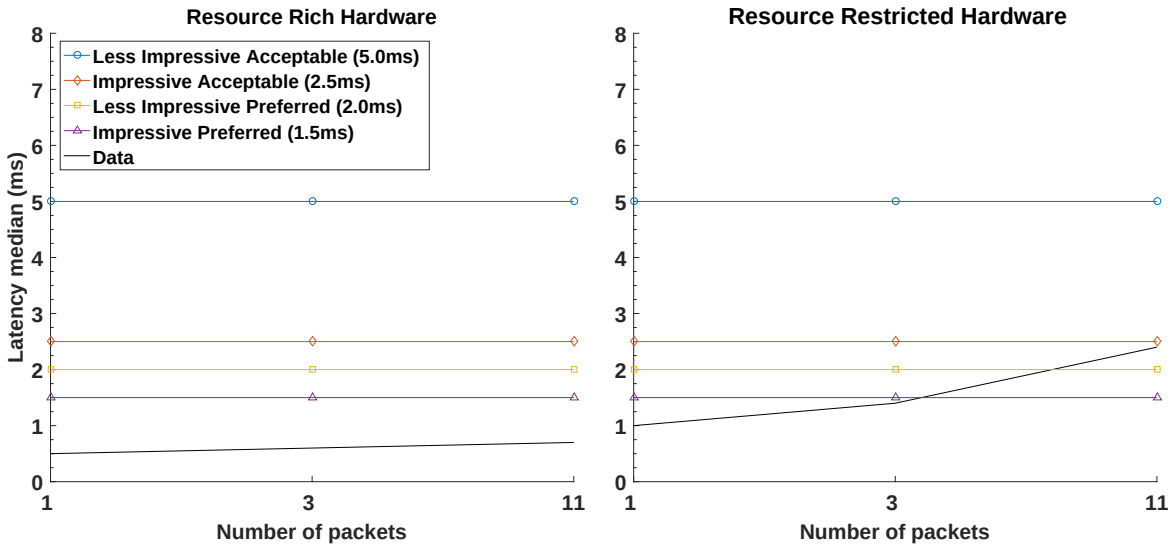

Figure 4.6: The median latency of OSs Ubuntu 16.04 ⟷ Ubuntu is shown on the left and the median latency of OSs Ubuntu 16.04 ⟷ Raspbian on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

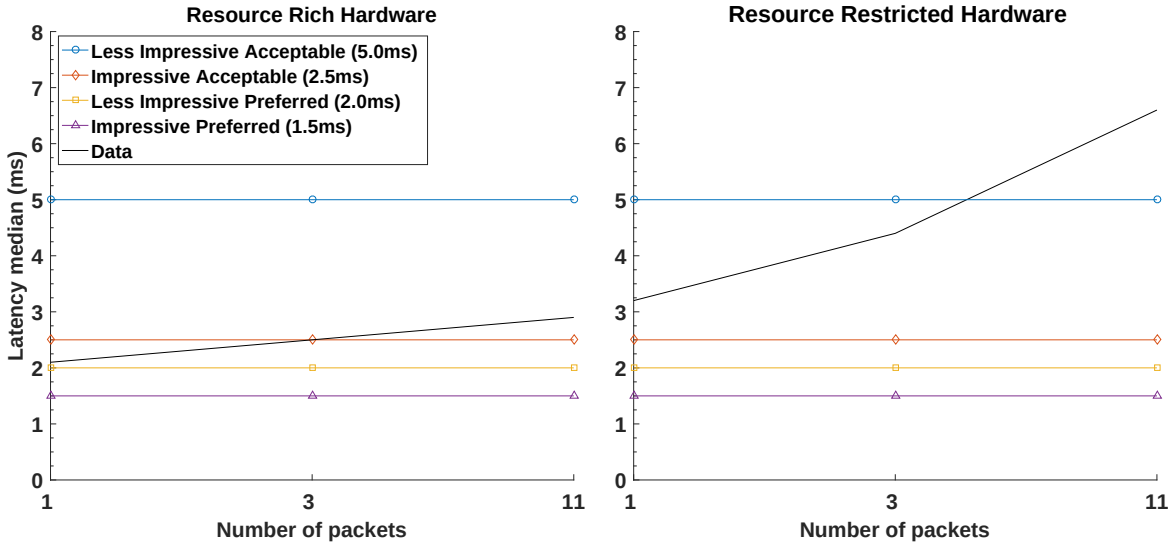

Figure 4.7: The median latency of OSs Windows 10 ⟷ Ubuntu 16.04 is shown on the left and the median latency of OSs Windows 10 ⟷ Raspbian is shown on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

impressive acceptable, **less impressive preferred**, and **impressive preferred** latency requirements was 61.2%, 0%, and 0% respectively.

The overall probability of meeting the **impressive acceptable**, **less impressive preferred**, and **impressive preferred** latency requirements was 0.0% when sending data between a resource-rich hardware with OS Windows 10 and a resource-restricted hardware with OS Raspbian. This also applied to sending data between a resource-rich hardware with OS Windows 10 and a resource-rich hardware with OS Ubuntu 16.04, except for the probability of meeting the **impressive acceptable** latency requirement. With a data size below 4,096 bytes the probability was 42.2% and higher, and the probability of meeting the **less impressive preferred** latency requirement with a data size below 1,024 bytes was 13.2%.

For detailed results, see Appendix A, Table A.4. This table presents the median, standard deviation, worst case, samples received, and probability of meeting the latency requirements under DDS QoS Reliability 'best effort'; the combinations of OS and hardware stated earlier; and one (equivalent to 4, 16, 64, 256, or 1,024 bytes data size), three (equivalent to 4,096 bytes data size), and 11 (equivalent to 16,384 bytes data size) packets.

Chapter 5

Discussion

5.1 Review of the Findings

The results presented in Chapter 4 are discussed in this section. First, a global discussion of the results will be presented. Second, the main research question of whether Data Distribution Service (DDS) can be used to reliably teleoperate robotic systems will be discussed. 'Reliably' refers to the research sub-question of whether DDS meets the latency requirements when subjected to different data sizes (ranging from 4 to 16,384 bytes) and DDS Quality of Service (QoS) Reliability settings 'reliable' and 'best effort' used under different operating conditions. These operating conditions are packet loss, network overhead, and Operating System (OS) and hardware, and are discussed with respect to the main question and sub-questions in Sections 5.1.1, 5.1.2, and 5.1.3.

5.1.1 Results of Introducing Packet Loss

The results of introducing different percentages of packet loss and varying data sizes indicate that when using DDS QoS Reliability 'reliable', the overall latency is higher than when using DDS QoS Reliability 'best effort' (see Section 4.1, Figure 4.1). This was expected, as DDS QoS Reliability 'best effort' does not resend lost data and DDS QoS Reliability 'reliable' does. For DDS QoS Reliability 'best effort', the average latency remains the same for different percentages of packet loss (Section 4.1, Figure 4.2). These average latencies take into account that lost data, which have an infinite latency, are not included in the results to measure the average latency. Including them would make the average latency infinite, and therefore unusable.

It was also expected that when using DDS QoS Reliability 'reliable', there would be a higher percentage of data received than when using DDS QoS Reliability 'best effort', since the 'reliable' setting resends lost data. The results given in Section 4.1, Figure 4.1 concluded otherwise. This could have been caused by an early shutdown of the receiver, before all samples were received. The shutdown of the receivers are linked to the shutdown of the transmitter. The transmitter shuts down when it is finished sending all 10,000 samples. The increased latency due to resending data shifts the arrival times of data. In this case, a part of the data arrives after the shutdown. Another possibility is that the introduced packet loss also affected the data being resend. This could have led to multiple resends of the same data, increasing latency and arrival times even further.

As mentioned in Section 2.4, data is divided into packets and then transferred over the network. The required number of packets can be computed by dividing the size of the data to be transferred by the Maximum Transmission Unit (MTU), which was 1,500 bytes during the tests, and than rounding it up to the nearest integer. For example, a data size of 4,096 bytes will be divided over three packets. If one or more of these packets are lost, the whole data set is regarded as lost. In other words, all three packets need to arrive in order to count as data received.

Each packet has the same probability of being lost during the tests. This means that at any packet loss percentage greater than 0%, data that is divided over multiple packets has a lower data receive rate compared to data that require only one packet. The expected percentage of data received at a given packet loss, data size, and MTU can be computed using Equation 5.1. This equation only applies to tests using DDS QoS Reliability 'best effort', as DDS QoS Reliability 'reliable' resends missing packets and should therefore have no data loss.

$$Data\_Received = \left(\frac{100 - packet\_loss}{100}\right)^{\left\lceil \frac{data\_size}{MTU} \right\rceil} \tag{5.1}$$

In conclusion, data with a smaller size than the MTU has the highest chance of being received when there is a chance of packet loss. Therefore, it is recommended to divide data that has a size greater than the MTU into smaller pieces of data with a size smaller than the MTU.

Conclusion

DDS QoS Reliability 'reliable' cannot be used to reliably teleoperate robotic systems used by the National Police of the Netherlands (NPN). As stated in Section 4.1, using DDS QoS Reliability 'reliable' and introducing a packet loss of 8% or higher results in a 4.9% or lower probability (sample receive rate (%) * latency requirement (%)) of all data sizes meeting the **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)** latency requirements. This means that even though data is resend, it is still not usable due to the latency exceeding the latency requirement.

DDS QoS Reliability 'best effort' also cannot be used to reliably teleoperate robotic systems used by the NPN. The probability to meet the requirement within time is 89.9% at the most optimum situation besides no packet loss (8% packet loss sending 1 data packet). At greater data sizes or packet loss percentages this probability is even lower. It is not possible to increase the probability by increasing the number of samples send. This is because the latency spikes are grouped and randomly distributed (see Section 5.2.1). For instance, a system requires 100 samples per minute to function and has a packet loss of 20%. Increasing the transmitted samples to 200 does not work even though 160 is more than the system requires to function properly. Because increasing the number of samples send from 100 to 200 per minute due to a packet loss of 20% still results in 20% of the samples randomly after one another missing the deadline, in other words a prolonged time of unusable information, resulting in an unreliable system.

To sum up, when there is an 8% or higher packet loss and DDS QoS Reliability 'reliable' or 'best effort' is set, DDS cannot be used to reliably teleoperate robotic systems used by the NPN. However, for lower packet loss percentages the reliability increases, since both sample received percentage and meeting the impressive and less impressive latency requirements increases up to 100%.

5.1.2 Results of Introducing Network Overhead

The results of introducing different percentages of network overhead and varying data sizes show that the latency and percentage data received remained the same when the used bandwidth was less than the maximum bandwidth. However, when the used bandwidth exceeded the maximum bandwidth the latency and packet loss increased. For this test setup, the required bandwidths were 99.6%, 98.3%, and 93.2% in order to exceed the maximum bandwidth for package sizes one, three, and 11 respectively. Compared with the test results shown in Section 4.2 (Figures 4.3 and 4.4), the latency increases and packets received decreases at the computed required bandwidth percentages (93.2%, 98.3%, and 99.6% with a $\approx 1\%$ margin). When the used bandwidth is equal to or higher than the maximum bandwidth, the latency increases and the sample receive rate decreases. Therefore, it is recommended to have the used bandwidth take up at most 99% of the maximum bandwidth.

Conclusion

DDS can be used to reliably teleoperate robotic systems used by the NPN for less impressive tasks when the used bandwidth is less than the maximum bandwidth, as the probability of meeting this requirement is 97.6% or higher. This also applies to impressive tasks with a data size up to 4,096 bytes, as the probability of meeting the **impressive acceptable (2.5 ms)** latency requirement is over 96.0%. However, when sending data with a data size of 16,384 bytes or greater DDS cannot be used to reliably teleoperate robotic systems during impressive tasks, because the probability of meeting the **impressive acceptable** latency requirement is 61.2% and lower.

DDS also cannot be used to reliably teleoperate robotic systems used by the NPN when the used bandwidth exceeds the maximum bandwidth. The probability of meeting the **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)** latency requirements are $\approx 0\%$.

5.1.3 Results of Using Different Operating Systems and Hardware

The results of using different OSs, as presented in Section 4.3, Figure 4.5, show that the latency median of OS Ubuntu 16.04 has the lowest average latency, followed by Raspbian and Windows 10. This was expected, because Windows 10 had 54 background programs running during the tests, whereas Ubuntu 16.04 had only 17. These latencies were measured using the same hardware with minimum active programs for both OSs. Also, the results of ping [15], which measures the latency using a different protocol, measured a higher average latency on Windows 10 compared to Ubuntu 16.04. The average measured latency for OS Ubuntu was 0.466 ms, whereas the average measured latency for OS Windows 10 was almost twice as high (0.84 ms).

The results of using resource-rich hardware versus resource-restricted hardware, as presented in Section 4.3 (Figures 4.6 and 4.7), indicate that the latency median is lower when using resource-rich hardware than when using resource-restricted hardware. This was expected, since resource rich-hardware has more resources available than resource-restricted hardware and therefore can handle data (packets) faster.

Conclusion

DDS can be used to reliably teleoperate robotic systems used by the NPN with OSs Ubuntu 16.04 and Raspbian in most situations. The only situation in which DDS cannot be used to reliably teleoperate robotic systems is for impressive tasks when distributing data between OS Ubuntu 16.04 and OS Raspbian and using a data size of 16,384 bytes. The probability of meeting the **impressive acceptable (2.5 ms)** and **impressive preferred (1.5 ms)** latency requirements are 61.2% and 0% respectively.

In most situations, DDS cannot be used to reliably teleoperate robotic systems used by the NPN when one of the OSs is Windows 10, except for less impressive tasks, for which the probability of meeting the **less impressive acceptable (5.0 ms)** latency requirement is above 99.3%.

Resource-rich hardware and resource-restricted hardware can both be used to reliably teleoperate robotic systems. However, it is recommended to use resource-rich hardware over resource-restricted hardware, since resource-rich hardware has the lowest median latencies ($\simeq 0.5\ ms$, $\simeq 0.6\ ms$, $\simeq 0.7\ ms$) and highest probability ($\simeq 100\%$) of meeting the latency requirements for one (equivalent to 4, 16, 64, 256, or 1,024 bytes data size), three (equivalent to 4,096 bytes data size), and 11 (equivalent to 16,384 bytes data size) packets.

5.2 Other Findings

5.2.1 Grouped Latency Spikes

The test results shown in Figure 5.1 (and a zoomed in version in Figure 5.2), showed that when the latency spiked it was followed by more consecutive but decreasing latency spikes over time. In this figure, the circled latency spike has two out of 10 data samples exceeding the **impressive acceptable (75 ms)** latency requirement.

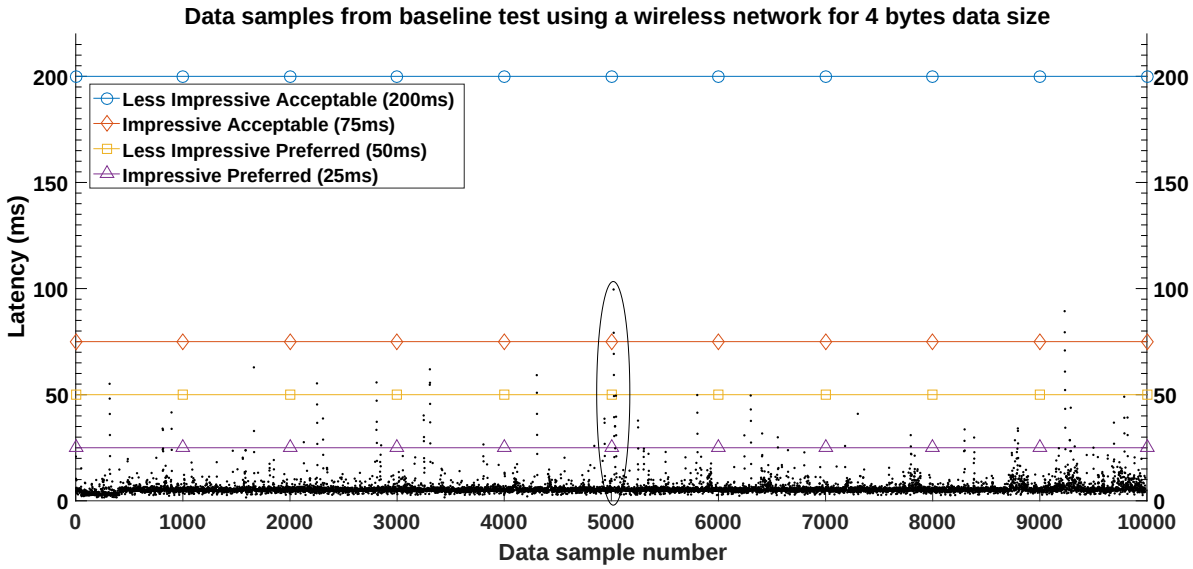

Figure 5.1: The latency for every data sample of the baseline test using a wireless network, along with the four latency requirements. The following operating conditions were used: 4 bytes data size, 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

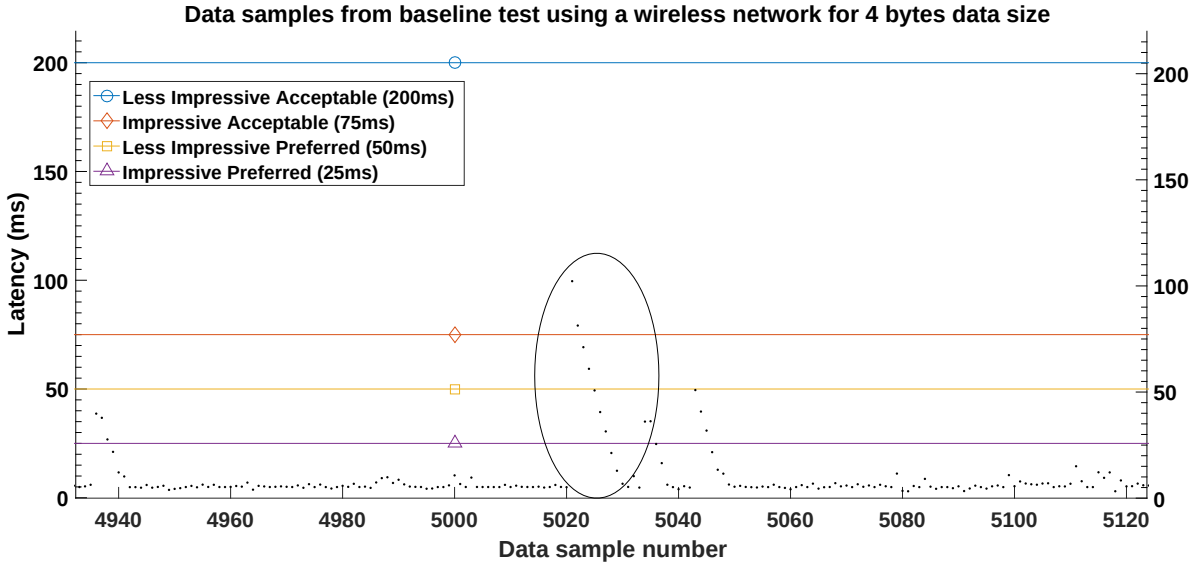

Figure 5.2: The latency for less data sample of the baseline test over a wireless network so that the grouped latency spikes are visible (circled), along with the four latency requirements. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

The latency spikes show no direct correlation between the distance of each consecutive data sample. These grouped latency spikes could harm the system when more than accounted for are exceeding the latency requirement. More precisely, when the latency of multiple frames or samples exceed the

latency requirement, the required frames per second (FPS), or frequency, cannot be met. For example, the required FPS, or receive frequency, in order to control the system in a reliable way is 10 frames or samples per second. If the system sends 10 samples or frames per second and two out of 10 have a latency exceeding the latency requirement and can therefore not be used, the required FPS, or frequency, cannot be met. Not meeting the FPS, or frequency, requirement means that there is a lack of information or actions, which could lead to high-level mission failure.

5.3 Conclusion

The main research question of whether DDS can be used to reliably teleoperate robotic systems used by the NPN under specified operating conditions is answered in this section. For this research, 'reliably' referred to having a high ($\simeq 100\%$) probability of meeting the latency requirements. In order to answer the research question, it was divided into three sub-questions: Does DDS meet the latency requirements when subjected to different data sizes (ranging from 4 to 16,384 bytes) and different types of DDS QoS settings (DDS QoS Reliability settings 'reliable' and 'best effort') under the following operating conditions:

- packet loss,
- network overhead,
- different OS and hardware configurations.

The NPN tasks were categorized as impressive and less impressive, each with their own latency requirements: **less impressive acceptable (200 ms)**, **impressive acceptable (75 ms)**, **less impressive preferred (50 ms)**, and **impressive preferred (25 ms)**.

The robotic systems are usually teleoperated over a wireless network. However, it was better to use a wired network for testing purposes because a wired network is more consistent and less sensitive to noise than a wireless network. The signal-to-noise ratio (SNR) for a wired network is constant and low, meaning that the results produced during the tests are only affected by the user's input. Therefore, for this study it was decided that a wired network would be used.

The latency of a wired network was 5 ms to 75 ms lower than the latency of a wireless network. Therefore, the latency requirements were scaled accordingly. The latency requirements for a wired network were scaled down to: **less impressive acceptable (5.0 ms)**, **impressive acceptable (2.5 ms)**, **less impressive preferred (2.0 ms)**, and **impressive preferred (1.5 ms)**.

The results of validating DDS with the operating condition of packet loss (Section 5.1.1) indicate that DDS cannot be used to reliably teleoperate robotic systems used by the NPN with a packet loss of 8% or higher for both DDS QoS Reliability 'reliable' and 'best effort'. However, DDS can be used to reliably teleoperate robotic systems used by the NPN when packet loss is ≈ 2%, and 1,026 bytes data size and DDS QoS Reliability 'best effort' are used. This will result in ≈ 98% reliability.

The results of validating DDS with the operating condition of network overhead indicate (see Section 5.1.2) that DDS can be used to reliably teleoperate robotic systems used by the police for less impressive tasks when the used bandwidth is less than the maximum bandwidth. This also applies to impressive tasks with a data size of up to 4,096 bytes. It does not apply to impressive tasks with a data size of 16,384 bytes. DDS cannot be used to reliably teleoperate robotic systems used by the NPN when the used bandwidth exceeds the maximum bandwidth.

The results of validating DDS with the operating condition of different OS and hardware configurations (Section 5.1.3) indicate that DDS can be used to reliably teleoperate robotic systems used by the NPN with OS Ubuntu 16.04 and Raspbian in most situations. The only situation in which DDS cannot be used to reliably teleoperate robotic systems is for impressive tasks when distributing data between OS Ubuntu 16.04 and OS Raspbian using a data size of 16,384 bytes. Also, DDS cannot be used to reliably teleoperate robotic systems used by the NPN when one of the OSs used is Windows 10, except for less impressive tasks. Resource-rich hardware and resource-restricted hardware can both be used to reliably teleoperate robotic systems.

5.4 Limitations

The limitations of this research in regards to the implementation of DDS, latency requirements, and conversion are discussed in this section.

For this research, DDS was not directly validated. Rather, it was validated through an implementation of DDS, namely OpenDDS. OpenDDS uses all the components of DDS but also adds its own component: DCPSInfoRepo, as illustrated in Section 2.3.2, Figure 2.7. This component detects when subscriptions and publications in a domain should be associated and notifies the Publisher and Subscriber to make the associations. This means that the latency at the start of the tests could be affected by the master's Publisher making an association with the slave's Subscriber, and vice versa. The latency during the tests should not be affected by this.

For the robotic systems controlled by the NPN, each police officer has a different latency requirement to perform their task with a high success rate. Some can still successfully perform their task with a high latency, whereas others require a very low latency. The latency requirements from other studies and those used for this research are therefore not applicable to every police officer. For example, some police officers can still successfully perform their impressive task with 200 ms latency, whereas for this research the **impressive preferred** and **impressive acceptable** latency requirements were determined to be 25 ms and 75 ms respectively. The latency requirements used for this research should be interpreted as guidelines rather than rules.

The robotic systems are usually teleoperated over a wireless network. However, for testing purposes it was better to use a wired network because a wired network is more consistent and less sensitive to noise. The method that was used to convert the results of a wired network to a wireless network equivalency involved scaling the latency requirements (Section 3.2). The latency requirements were scaled twice, and thus induced possible errors twice. However, the latency requirements for the wireless network should be used as a guideline and are therefore flexible with room for errors.

5.5 Recommendations

For this project, most of the possible operating conditions were tested. However, there are still some operating conditions that future research could examine. Some of these operating conditions could potentially increase the probability of meeting latency requirements.

- In the case that multiple data is transferred, DDS QoS transport priority can be used to prioritize important data over less important data. This will likely affect the latency and should therefore be tested in future studies.
- The wireless baseline test was performed using WiFi. Future research could test a wireless network alternative, for example, 4G.

In addition to these recommendations regarding future work, several recommendations could be made regarding the results discussed in Section 5.1. These recommendations are presented below.

- Data with a size smaller than the MTU has the highest chance of being received when there is a chance of packet loss. Therefore, it is recommended to divide data, that has a size greater than the MTU over pieces of data with a size smaller than the MTU.
- When the used bandwidth is equal to or higher than the maximum bandwidth, the latency increases and the sample receive rate decreases. Therefore, it is recommended to have the used bandwidth take up at most 99% of the maximum bandwidth.
- The OS Ubuntu 16.04 had the lowest median latency of the three tested OSs (Ubuntu 16.04, Windows 10, Raspbian) and the highest probability of meeting the latency requirements. OS Windows 10 had the highest median latency and the lowest probability of meeting the latency requirements, especially for the impressive latency requirements. Therefore, it is recommended that OS Ubuntu 16.04 or Raspbian is used, and OS Windows 10 is avoided.
- Resource-rich hardware and resource-restricted hardware can both be used to reliably teleoperate robotic systems. However, it is recommended that resource-rich hardware is used over resource-restricted hardware, as resource-rich hardware had the lowest median latency ($\simeq 0.5\ ms$, $\simeq 0.6\ ms$, $\simeq 0.7\ ms$) and the highest probability ($\simeq 100\%$) of meeting the latency requirements.

List of Acronyms

DDS Data Distribution Service
RTI Real-Time Innovations
RaM Robotics and Mechatronics
UDP User Datagram Protocol
Hz Hertz
QoS Quality of Service
MTU Maximum Transmission Unit
OS Operating System
SNR signal-to-noise ratio
BER bit error ratio
OMG Object Management Group
RoVe Robots for Security
NPN National Police of the Netherlands
FPS frames per second
RTT round-trip time

Appendix A

Results

More detailed test results from the baseline tests, latency spikes, packet loss, network overhead, and Operating System (OS) and hardware are presented in this appendix. The relation between the number of packets and latency was greater than the relation between data size and latency. Therefore, the results are displayed inside the tables over number of packets instead of data size. This also helps to provide a clearer overview of the test results, as some tables contain a lot of information.

A.1 Baseline Results

		Data size (bytes)						
		4	16	64	256	1024	4096	16384
Median	Wireless (ms)	5.1	5.4	5.6	6.0	5.6	5.5	70.1
	Wired (ms)	0.9	1.0	1.0	1.0	1.1	1.4	2.4
Difference	-	4.2	4.4	4.5	4.9	4.5	4.1	67.6
Worst case	Wireless (ms)	99.6	65.8	73.5	85.8	134.2	261.2	363.1
	Wired (ms)	5.1	5.4	5.7	2.6	4.0	5.4	3.8
Difference	-	94.5	60.4	67.8	83.2	130.2	255.8	359.3

Table A.1: Wired and wireless baseline test results showing the median, the worst case or maximum latency, and the difference between the wired and wireless networks' medians and maximums.

A.2 Packet Loss Results

QoS	P.L. (%)	P	Med. (ms)	Std. (ms)	WC (ms)	Samp. Rec. (%)	Imp. Pref.	Less Imp. Pref.	Imp. Accept.	Less Imp. Accept.
Reliable	0	1	1.1	0.2	9.1	100.0	96.4	98.2	99.2	100.0
		3	1.5	0.3	6.0	100.0	5.8	94.0	94.8	100.0
		11	2.6	0.2	8.8	100.0	0.0	0.0	0.2	99.9
	8	1	381.7	364.2	2402.9	49.4	9.1	9.5	9.5	9.8
		3	6699.1	4684.9	16503.3	38.6	0.0	0.0	0.0	0.0
		11	50159.3	28013.0	94639.1	5.5	0.0	0.0	0.0	0.0
	16	1	722.0	501.0	3652.1	43.3	0.5	0.5	0.5	0.5
		3	26069.4	12327.7	45565.8	24.2	0.0	0.1	0.1	0.2
		11	71500.8	26492.1	99222.1	1.4	0.0	0.0	0.0	0.0
Best Effort	0	1	1.0	0.1	5.7	100.0	99.4	99.9	100.0	100.0
		3	1.4	0.1	5.4	100.0	97.0	99.9	100.0	100.0
		11	2.4	0.1	3.8	100.0	0.0	0.0	61.2	100.0
	8	1	1.0	0.2	6.7	92.1	97.6	98.8	99.8	100.0
		3	1.6	0.2	8.1	78.7	73.2	97.9	99.5	100.0
		11	2.5	0.2	7.1	36.1	0.0	0.0	6.8	100.0
	16	1	1.0	0.2	7.1	84.3	94.6	99.4	99.8	100.0
		3	1.5	0.2	3.6	58.0	59.6	95.8	99.6	100.0
		11	2.5	0.3	6.9	11.9	0.0	0.0	6.8	99.7

Table A.2: The measured and computed median (Med.), standard deviation (Std.), worst case or maximum latency (WC), information or samples received (Samp. Rec.), and the probability of meeting the impressive (Imp.) and less impressive (Less Imp.) preferred (Pref.) and acceptable (Accept.) latency requirements at different numbers of packets (P), different percentages of packet loss (P.L.), and different Data Distribution Service (DDS) Quality of Services (QoSs).

A.3 Network Overhead Results

QoS	N (%)	P	Med. (ms)	Std. (ms)	WC (ms)	Samp. Rec. (%)	Imp. Pref.	Less Imp. Pref.	Imp. Accept.	less Imp. Accept.
Reliable	0	1	1.1	0.2	9.1	100.0	96.4	98.2	99.3	100.0
		3	1.5	0.3	5.6	100.0	5.8	94.0	94.8	100.0
		11	2.6	0.2	8.8	100.0	0.0	0.0	0.2	99.9
	50	1	1.2	0.2	6.4	100.0	94.8	98.1	98.8	100.0
		3	1.7	0.3	5.3	100.0	0.1	88.5	97.5	99.9
		11	2.9	0.2	9.3	100.0	0.0	0.0	0.0	99.8
	95	1	1.4	0.4	8.3	100.0	70.7	96.5	97.7	99.7
		3	1.9	0.4	7.9	100.0	0.1	88.5	96.2	99.3
		11	4.4	8914.0	35660.3	0.2	0.0	0.0	0.0	67.8
Best Effort	0	1	1.0	0.1	5.7	100.0	99.4	99.9	100.0	100.0
		3	1.4	0.1	5.4	100.0	97.0	99.9	100.0	100.0
		11	2.4	0.1	3.8	100.0	0.0	0.0	61.2	100.0
	50	1	1.1	0.2	14.6	100.0	97.2	98.9	99.8	100.0
		3	1.5	0.1	6.4	100.0	50.7	98.8	99.7	100.0
		11	2.8	0.2	4.4	100.0	0.0	0.0	0.0	100.0
	90	1	1.3	0.3	9.0	100.0	87.3	97.3	98.5	100.0
		3	1.8	0.2	7.5	100.0	8.2	95.6	98.1	100.0
		11	2.6	0.3	8.7	100.0	0.0	0.0	0.0	99.7
	92	1	1.3	0.3	9.3	100.0	88.8	97.8	98.8	100.0
		3	1.8	0.2	5.5	100.0	18.0	94.3	99.5	100.0
		11	2.9	0.6	9.5	100.0	0.0	0.0	0.0	97.6
	94	1	1.3	0.2	10.8	100.0	91.1	98.2	99.1	100.0
		3	1.8	0.3	7.2	100.0	8.9	92.8	96.0	99.9
		11	5.8	0.3	7.7	12.0	0.0	0.0	0.0	1.8
	95	1	1.3	0.3	9.9	100.0	86.0	96.8	98.5	100.0
		3	1.5	0.2	5.9	100.0	41.1	97.0	99.3	100.0
		11	6.1	0.5	10.5	3.7	0.0	0.0	0.0	2.5
	96	1	1.4	0.2	9.4	100.0	80.8	99.0	99.7	100.0
		3	1.5	0.2	5.0	100.0	19.8	97.0	99.3	100.0
		11	6.0	1.0	7.1	0.3	0.0	0.0	0.0	24.1
	97	1	1.3	0.3	7.8	100.0	88.6	97.9	98.9	100.0
		3	1.5	0.2	4.8	100.0	42.4	97.7	99.4	100.0
		11	6.2	1.3	7.4	0.2	0.0	0.0	0.0	40.0
	98	1	1.4	0.3	5.2	100.0	68.5	97.0	98.9	100.0
		3	5.2	0.3	8.8	100.0	0.0	0.0	0.0	43.8
		11	4.9	1.2	6.6	0.1	0.0	0.0	0.0	55.6
	99	1	3.2	0.6	10.4	98.1	5.8	25.4	39.9	82.3
		3	5.4	0.8	6.1	1.0	0.0	0.0	0.0	28.7
		11	5.4	1.0	6.7	0.1	0.0	0.0	0.0	33.3

Table A.3: The measured and computed medians (Med.), standard deviations (Std.), worst cases or maximum values (WC), information or samples received (Samp. Rec.), and probability of meeting the impressive (Imp.) preferred (Pref.) and acceptable (Accept.) and less impressive (Less Imp.) preferred and acceptable latency requirements at different numbers of packets (P), different network overhead percentages (N), and different DDS QoSs.

A.4 Operating Systems and Hardware Results

OS master ⟷ slave	P.	Med. (ms)	Std. (ms)	WC (ms)	Samp. Rec. (%)	Imp. Pref.	Less Imp. Pref.	Imp. Accept.	Less Imp. Accept.
Ubuntu ⟷ Ubuntu	1	0.5	0.2	48.3	100.0	99.5	99.9	99.9	100.0
	3	0.6	0.0	2.5	100.0	100.0	100.0	100.0	100.0
	11	0.7	0.1	4.4	100.0	99.9	100.0	100.0	100.0
Ubuntu ⟷ Raspbian	1	1.0	0.1	5.7	100.0	99.4	99.9	100.0	100.0
	3	1.4	0.1	5.4	100.0	97.0	99.9	100.0	100.0
	11	2.4	0.1	3.8	100.0	0.0	0.0	61.2	100.0
Windows 10 ⟷ Ubuntu	1	2.1	0.3	31.7	100.0	0.0	13.2	97.2	99.8
	3	2.5	0.3	11.9	100.0	0.0	0.0	42.2	99.7
	11	2.9	0.6	32.7	100.0	0.0	0.0	0.0	99.6
Windows 10 ⟷ Raspbian	1	3.2	0.5	30.7	100.0	0.0	0.0	0.0	99.3
	3	4.4	0.2	9.3	100.0	0.0	0.0	0.0	99.6
	11	6.6	0.2	14.9	100.0	0.0	0.0	0.0	0.0

Table A.4: The OSs used for the master and the slave and the number of packets (P.) are on the vertical axes coloured in light grey. The median (Med.), standard deviation (Std.), worst case or maximum value (WC), percentage of samples received (Samp. Rec.), and the probability of meeting the impressive (Imp.) and less impressive (Less Imp.) preferred (Pref.) and acceptable (Accept.) latency requirements are on the horizontal axes coloured in dark grey.

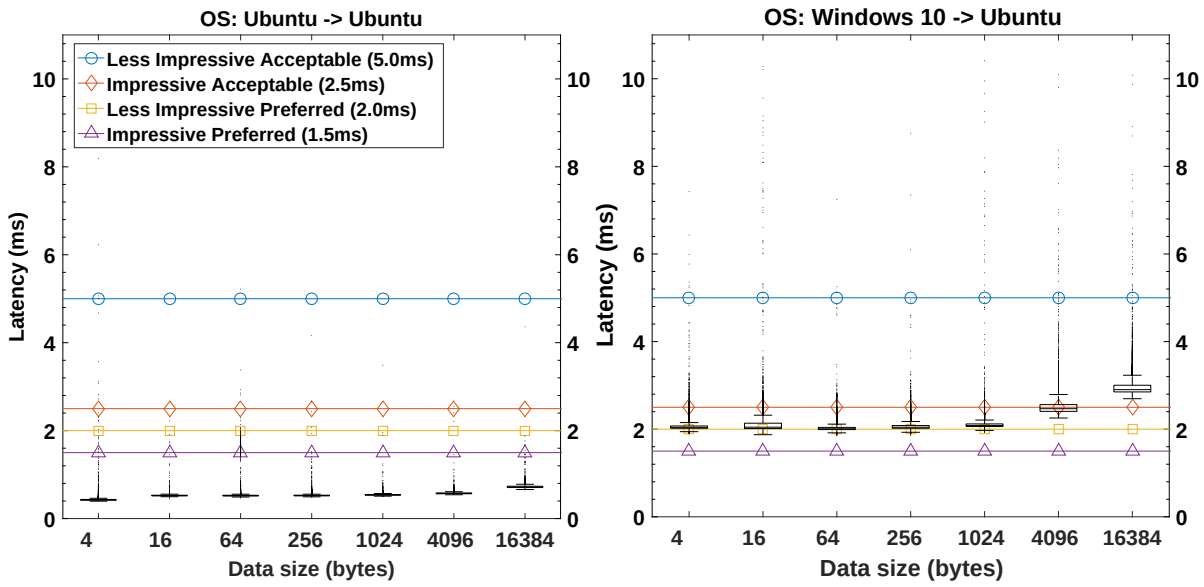

Figure A.1: The latency boxplot of OSs Ubuntu 16.04 ⟷ Ubuntu 16.04 is shown on the left and the latency boxplot of OS Windows 10 ⟷ Ubuntu 16.04 is shown on the right, along with the four latency requirements. In addition to these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

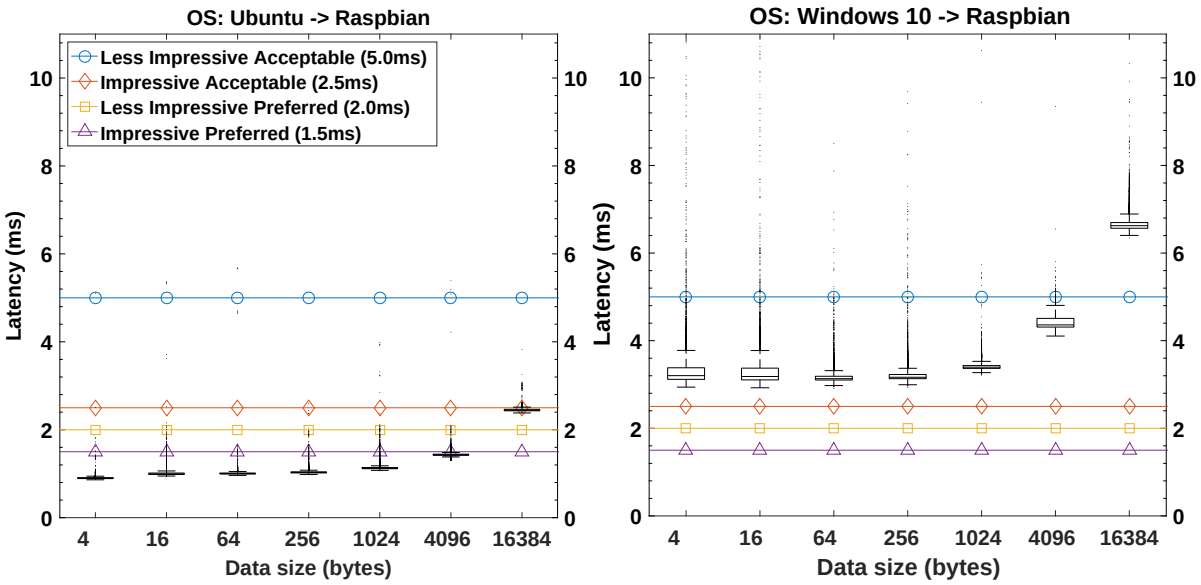

Figure A.2: The latency boxplot of the OSs Ubuntu 16.04 ⟷ Raspbian is shown on the left and the latency boxplot of the OSs Windows 10 ⟷ Raspbian is shown on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

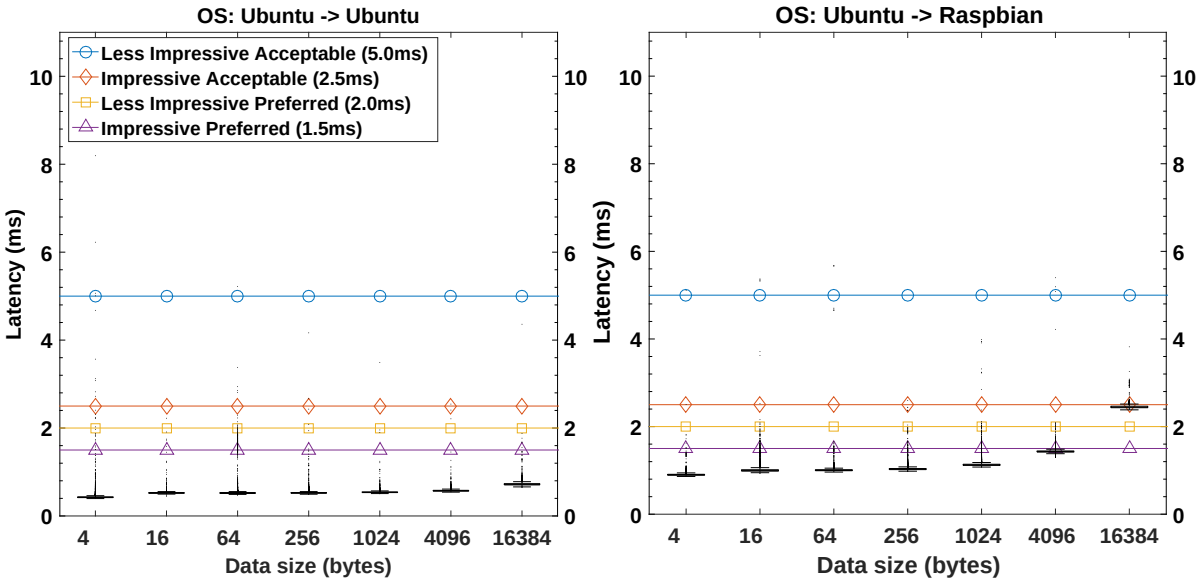

Figure A.3: The latency boxplot of the OSs Ubuntu 16.04 ⟷ Ubuntu 16.04 is shown on the left and the latency boxplot of the OSs Ubuntu 16.04 ⟷ Raspbian is shown on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

A.5 Latency Spikes Results

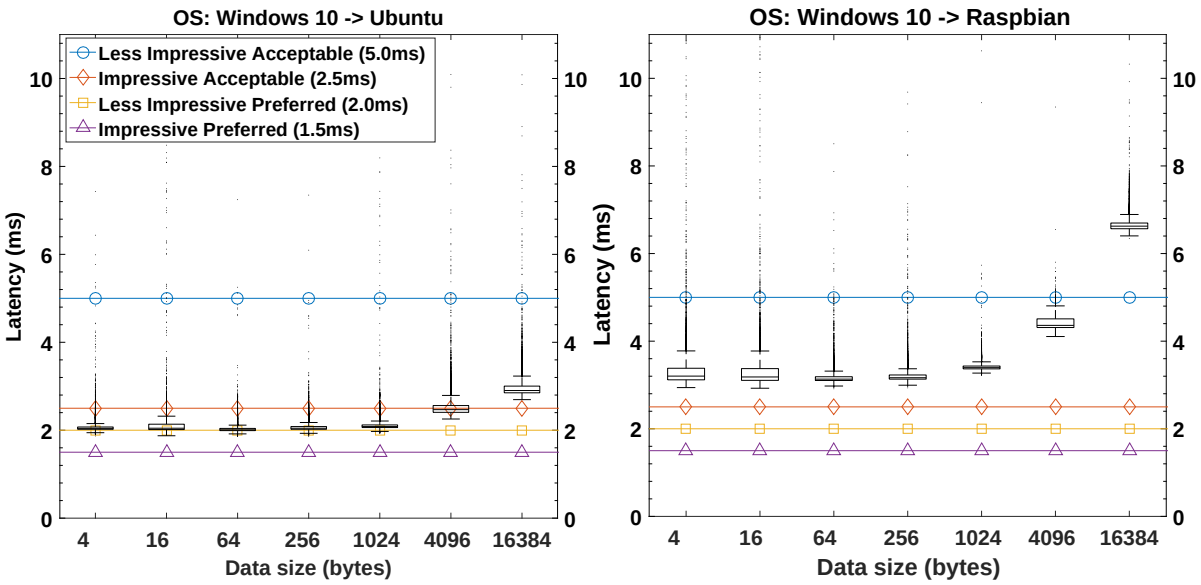

Figure A.4: The latency boxplot of the OSs Windows 10 ⟷ Ubuntu 16.04 is shown on the left and the latency boxplot of the OSs Windows 10 ⟷ Raspbian is shown on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and DDS QoS Reliability 'best effort'.

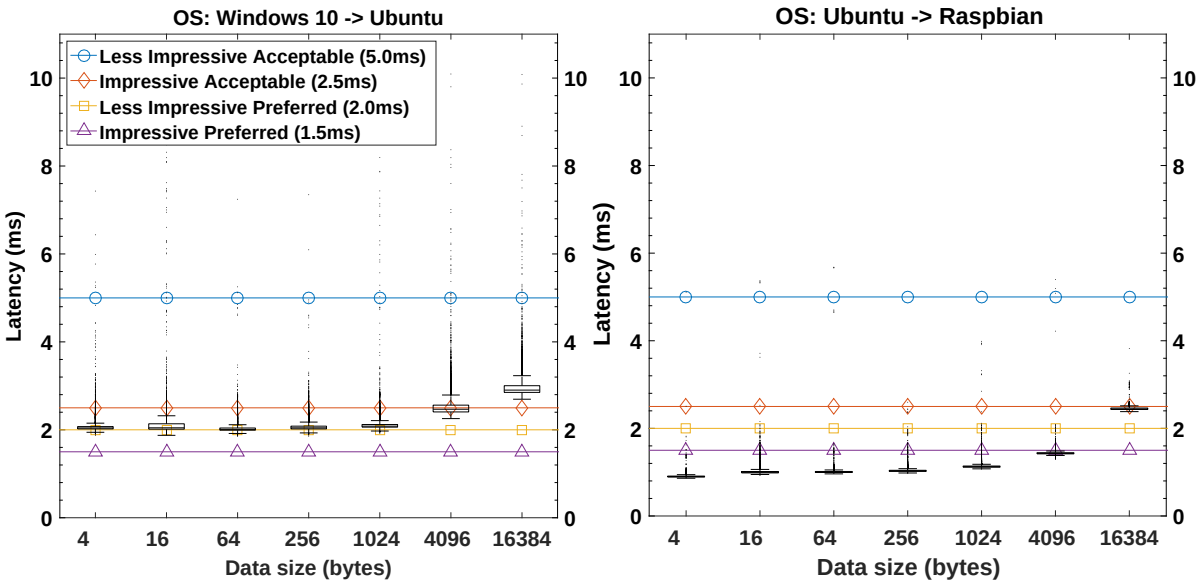

Figure A.5: The latency boxplot of OS Windows 10 ⟷ Ubuntu 16.04 is shown on the left and the latency boxplot of the OS Ubuntu 16.04 ⟷ Raspbian is shown on the right, along with the four latency requirements. Besides these OSs, the following operating conditions were used: 0% network overhead, 0% packet loss, and QoS Reliability 'best effort'.

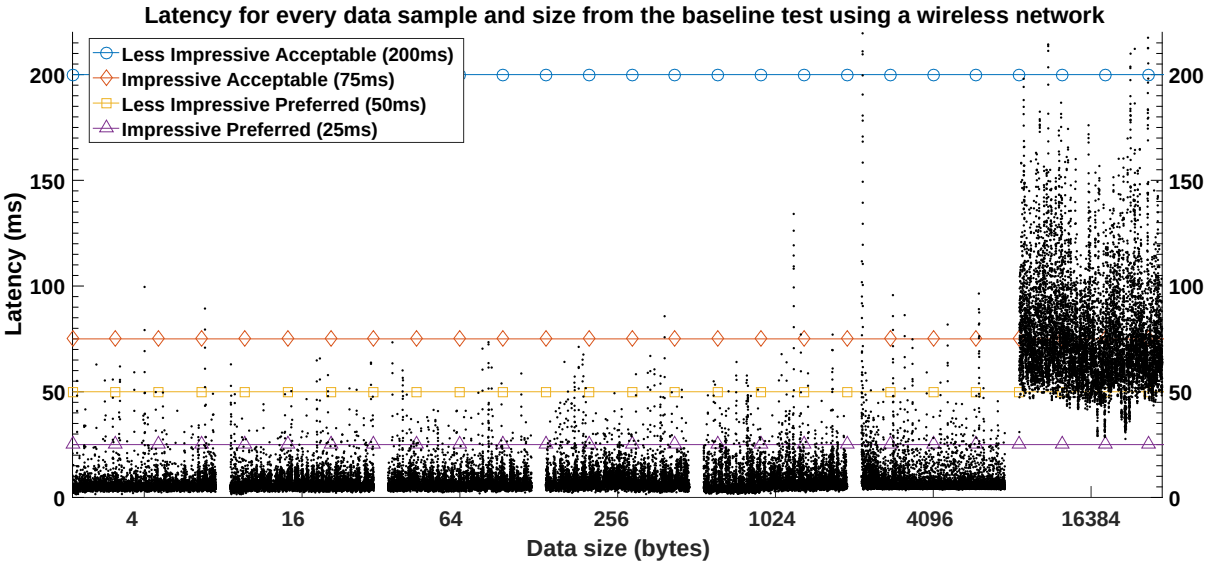

Figure A.6: The latency for every data sample and size of the baseline test over a wireless network along with the four latency requirements. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

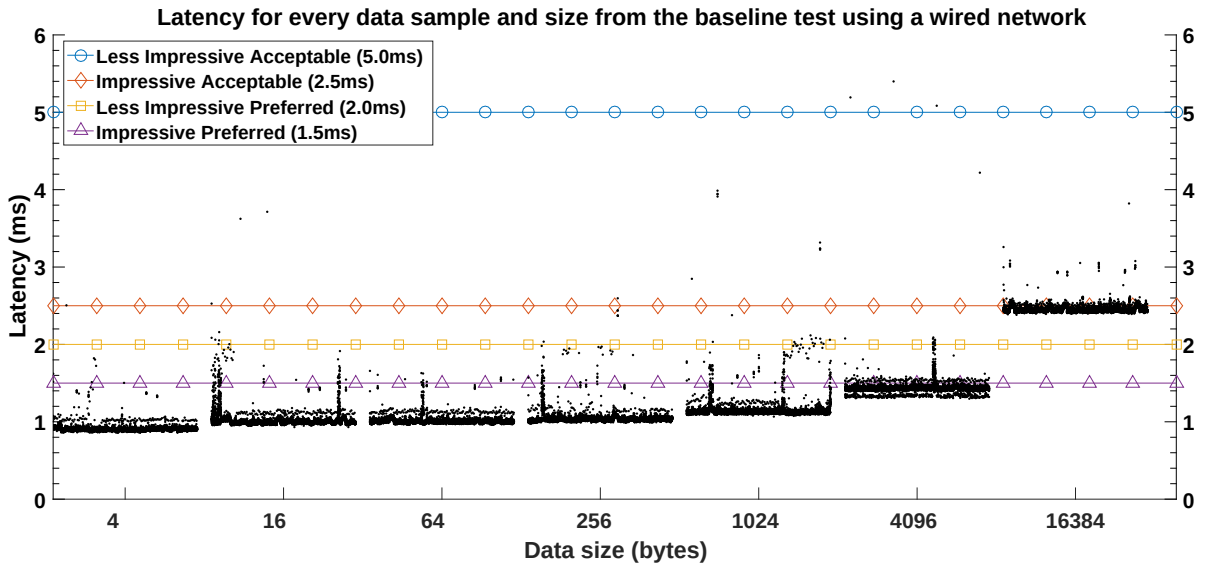

Figure A.7: The latency for every data sample and size of the baseline test over a wired network along with the four latency requirements. The following operating conditions were used: 0% packet loss, DDS QoS Reliability 'best effort', and OSs Ubuntu 16.04 ⟷ Raspbian.

Appendix B

User Manual

Checklist before starting the tests:

- Check that WiFi is disabled.
- Close unused programs (especially ones that use the Internet), such as Dropbox, Firefox, and virus scanners.
- Delete existing test results in order to watch progress.
- Check if the bash script is correct (it might have been changed for some minor tests).
- Check if the master/slave code has the desired functionality. Sometimes the functionally is added or disabled for more specific tests.

The following start sequence is used for every test:

1. Open the folder in which the test results will be placed in order to watch the test's progress. It is preferable to open the sub folder, since it generates the file corresponding with the current test whereas the pub folder generates the file after the current test is completed.
2. On the slave: open one terminal and set its path to the folder that contains all of the slave code.
3. Execute the bash script or command of slave subscriber.
4. On the master: open two terminals and set their path to the folder that contains all of the master code.
5. On one terminal: execute the bash script or command of master subscriber.
6. On the other terminal: execute the bash script or command of master publisher.

B.1 Automatic Testing

In order to run multiple tests automatically after one another, *Master Subscriber*, *Master Publisher* and *Slave Subscriber* have adjustable input parameters. The adjustable input parameters are: reliability, deadline (ns), latency (ns), priority, frequency (μs), data size ($32\ bits$), packet loss (%) and file name.

Fifteen bash scripts are created to automatically run the basic tests, saving time in the long run. The *Master Publisher*, *Master Subscriber* and *Slave Subscriber*, each have one bash script for packet loss and network overhead and three for different test setups.

Specific bash script command sequence for testing the operating condition *packet loss*, with Operating System (OS) Ubuntu 16.04 on the master and OS Raspbian on the slave:

- Slave:
  1. Subscriber: ./rtb_slave_1_PL
- Master:
  1. Subscriber: ./rtb_sub_1_PL
  2. Publisher: ./rtb_pub_1_PL

Specific bash script command sequence for testing the operating condition *network overhead*, with OS Ubuntu 16.04 on the master and OS Raspbian on the slave:

- Slave:
  1. separate terminal: iperf -s -u -i 1
     (a) first argument: indicates server

(b) second argument: indicates that it uses the data transfer protocol User Datagram Protocol (UDP).
(c) third and fourth argument: show every second the occupied bandwidth by iperf.

2. Subscriber: ./rtb_slave_2_NO 90 1
   - first argument: network overhead % for folder name
   - second argument: Quality of Service (QoS) Reliability, 1 = reliable and 2 = best effort

- Master:
  1. separate terminal: iperf -c 10.42.0.210 -u -t 10000000000 -b 86m -i 1
     (a) first argument: indicates its a client
     (b) second argument: IP address of slave
     (c) third argument: indicates that it uses the data transport protocol UDP
     (d) fourth and fifth argument: indicates the total amount of time in seconds to transfer data with a certain bandwidth.
     (e) sixth and seventh argument: the bandwidth to be occupied in megabits per second, in this case 86 megabits per second.
     (f) eighth and ninth argument: one second interval to show the current bandwidth
  2. Subscriber: ./rtb_sub_2_NO 90 1
     - first argument: current network overhead %, this is used for the folder name.
     - second argument: QoS Reliability, 1 = reliable and 2 = best effort
  3. Publisher: ./rtb_pub_2_NO 90 1
     - first argument: network overhead % for folder name
     - second argument: QoS Reliability. 1 = reliable and 2 = best effort

Specific bash script command sequence for testing the OS, with OS Ubuntu on the master and OS Ubuntu on the slave:

- Slave:
  1. Subscriber: ./rtb_slave_3_linux
- Master:
  1. Subscriber: ./rtb_sub_3_linux
  2. Publisher: ./rtb_pub_3_linux

Specific bash script command sequence for testing the OS, with OS Windows 10 on the master and OS Raspbian on the slave:

- Slave:
  1. Subscriber: ./rtb_slave_3_raspbian
- Master:
  1. Subscriber: rtb_sub_3_raspbian
  2. Publisher: rtb_pub_3_raspbian

Specific bash script command sequence for testing the OS, with OS Windows 10 on the master and OS Ubuntu on the slave:

- Slave:
  1. Subscriber: ./rtb_slave_3_windows
- Master:
  1. Subscriber: rtb_sub_3_linux
  2. Publisher: rtb_pub_3_linux

B.2 Manually Testing

There are a maximum of eight input parameters: reliability, deadline (ns), latency (ns), priority, frequency (μs), data size ($32bits$), packet loss (%) and file name. All are unsigned numbers except for file location/name. Reliability, deadline, latency and priority are Data Distribution Service (DDS) QoS, of which only reliability is used.

Master Publisher uses all input parameters. *Master Subscriber* uses only reliability, deadline, latency and file location/name. *Laptop Slave* uses only reliability, deadline, latency, data size and file location/name.

The following command example is used on *Master Subscriber*:

- ./subscriber -DCPSConfigFile rtps.ini 2 57142857 0 FILE_LOCATION/File_name.txt
    1. third argument: DDS QoS Reliability, 1 = reliable, 2 = best effort.
    2. fourth argument: Deadline (ns) is the time within a sample should arrive.
    3. fifth argument: DDS QoS Latency budget (ns).
    4. sixth argument: File location plus name to print the test results to.

The following command example is used on *Master Publisher*:

- ./publisher -DCPSConfigFile rtps.ini 2 57142857 0 0 20000 4 0 FILE_LOCATION/File_name.txt
    1. third argument: DDS QoS Reliability, 1 = reliable, 2 = best effort.
    2. fourth argument: Deadline (ns) is the time within a sample should arrive.
    3. fifth argument: DDS QoS Latency budget (ns).
    4. sixth argument: DDS QoS Priority.
    5. seventh argument: Frequency, or time between each consecutive data sample in μs.
    6. eighth argument: Data size, where the filled number is multiplied with 32 bits. In this example the number 4 stands for 4*32 = 128 bits or 16 bytes.
    7. ninth argument: the required Packet loss in percentages (%)
    8. tenth argument: File location plus name to print the test results to.

The following command example is used on *Slave Subscriber*:

- ./subscriber -DCPSConfigFile rtps.ini 1 57142857 0 4 FILE_LOCATION/File_name.txt
    1. third argument: DDS QoS Reliability, 1 = reliable, 2 = best effort.
    2. fourth argument: Deadline (ns) is the time within a sample should arrive.
    3. fifth argument: DDS QoS Latency budget (ns).
    4. sixth argument: Data size, where the filled number is multiplied with 32 bits. In this example the number 4 stands for 4*32 = 128 bits or 16 bytes.
    5. seventh argument: File location plus name to print the test results to.

Appendix C

Compiling Testbench

Instructions for building the code for *Master Subscriber*, *Master Publisher* and *Slave Subscriber* (assuming OpenDDS-3.11 or higher is installed with ACE, TAO, DDS, and MPC correctly installed and configured):

Step 1: Open up a new terminal and locate to the folder containing the main code of the master or the slave. The following steps include command lines that should be entered after one another.

Step 2: Generate the export file.

- On a Linux-based Operating System (OS):
  - `$ACE_ROOT/bin/generate_export_file.pl` TestBenchCommon > TestBench-Common_Export.h
- On a Windows-based OS:
  - `%ACE_ROOT%/bin/generate_export_file.pl` TestBenchCommon > TestBench-Common_Export.h

Step 3: Run Make Project Creator to generate build files.

- On a Linux-based OS:
  - `$ACE_ROOT/bin/mwc.pl` -type gnuace TestBench.mwc
- On a Windows-based OS:
  - perl `%ACE_ROOT%\bin\mwc.pl` -type vc14 TestBench.mwc

Step 4: Build the application.

- On a Linux-based OS:
  - make
- On a Windows-based OS:
  - TestBench.sln
  - Inside this environment right mouse click on "solution TestBench" and click "build solution"
  - PS: Add #include <sstream> when using sstreams (in publisher) and #include <ace/OS_NS_time.h> when using ACE_OS times (in DataReader)

Bibliography

[1] R. Andringa. Politie zet drone in boven verkeersongelukken. *Binnenland*, 2017.

[2] Bandwidth (computing). [online]. Available: https://en.wikipedia.org/wiki/Bandwidth_(computing). [Accessed: 2018-03-21].

[3] P. Bellavista, A. Corradi, L. Foschini, and A. Pernafini. Data distribution service (dds): A performance comparison of opensplice and rti implementations. pages 377–383, 2013. cited By 4.

[4] Bit error rate . [online]. Available: https://en.wikipedia.org/wiki/Bit_error_rate. [Accessed: 2018-03-22].

[5] B. S. Blanchard and W. J. Fabrycky. *Systems Engineering and Analysis*. Pearson, fifth edition, 2013.

[6] J. Y. C. Chen and J. E. Thropp. Review of low frame rate effects on human performance. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 37(6):1063–1076, 2007.

[7] M. Claypool and D. Finkel. The effects of latency on player performance in cloud-based games. In *Proceedings of the 13th Annual Workshop on Network and Systems Support for Games*, NetGames '14, pages 2:1–2:6, Piscataway, NJ, USA, 2014. IEEE Press.

[8] Data Distribution Service. [online]. Available: http://portals.omg.org/dds/. [Accessed: 2018-03-06].

[9] S. Haykin and M. Moher. *Introduction to Analog & Digital Communications*. Wiley, second edition, 2006.

[10] S. Jörg, A. Normoyle, and A. Safonova. How responsiveness affects players' perception in digital games. In *ACM Symposium on Applied Perception 2012, SAP '12, Los Angeles, CA, USA - August 03 - 04, 2012*, pages 33–38, 2012.

[11] Modulation technique. [online]. Available: https://en.wikipedia.org/wiki/Modulation. [Accessed: 2018-03-21].

[12] Network overhead. [online]. Available: https://en.wikipedia.org/wiki/Overhead_(computing). [Accessed: 2018-03-17].

[13] Object Management Group (OMG). Data Distribution Service for Real-time Systems, Version 1.2. OMG Document Number formal/07-01-01, Available: http://www.omg.org/cgi-bin/doc?formal/07-01-01. [Accessed: 2018-03-17].

[14] OpenDDS. [online]. Available: http://opendds.org/about/dds_overview.html. [Accessed: 2018-03-17].

[15] Ping (networking utility) . [online]. Available: https://en.wikipedia.org/wiki/Ping_(networking_utility). [Accessed: 2018-03-22].

[16] S. Stramigioli and K. J. Russcher. RoVe robots for security. Available: https://www.ram.ewi.utwente.nl/research/project/rove.html. [Accessed: 2018-03-06].

[17] Videoconferencing Traffic: Network Requirements . [online]. Available: https://community.jisc.ac.uk/library/videoconferencing-booking-service/videoconferencing-traffic-network-requirements. [Accessed: 2018-03-22].

[18] J. Yang, K. Sandström, T. Nolte, and M. Behnam. Data distribution service for industrial automation. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation, ETFA 2012, Krakow, Poland, September 17-21, 2012*, pages 1–8, 2012.