Title of project placed here

Chongbin Ren

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the
Degree of Master of Science at The University of Glasgow

Date of submiss

Abstract

abstract goes here

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

Acknowledgements

I would like to thank the following people for their help and support throughout the project.

Contents

Chapter 1

Introduction

1.1 Context

Chapter 2

Background

2.1 Robotics Overview

2.1.1 Multi-Robot Systems

2.1.2 Raspberry Pi

2.1.3 SunFounder Raspberry Pi Smart Video Car

2.2 Robot Software and Operating System

2.2.1 ROS

2.2.2 ROS 2

2.2.3 Ubuntu

2.3 Communication on ROS vs ROS2

2.3.1 Star Network Topology

2.3.2 TCP and UDP in ROS

2.3.3 Data Distribution Service (DDS) on ROS2

Chapter 3

Installation and configuration

3.1 Ubuntu and ROS system installation

3.1.1 Ubuntu installation

1. Download the latest version image of Ubuntu Bionic (64-bits) for Raspberry Pi from official website.

2. Write the image to Raspberry Pi 3B+ by using SD card.

3. Run it with screen and configure all necessary dependency and install ”openssh” server for remote control.

Issues and Solution

1. Other operating system like Raspbian 10(buster) is not well supported by ROS system. There are some outdated dependencies which is needed in installing ROS. Therefore, there are many bugs and manually installation steps to solve when user want to install ROS in Raspbian operating system. Then the Ubuntu is recommend operating system as official website said.

2. Find the 64-bits version operating system because ROS2 only can run on 64-bits operating system.

3.1.2 ROS1 Installation

1. Follow the step details at first appendix A1 ROS1.

2. Configure Ubuntu repositories to allow ”restricted,” ”universe,” and ”multiverse.”

3. After installation, use *roscore* to test whether ROS system exist.

3.1.3 ROS2 Installation

1. Installing ROS2 via Debian Packages.

2. Follow the step details at first appendix A2 ROS2.

3. Using *ros2 run XXpackage XXfile* to test whether ROS2 can run successfully.

3.2 Router network configuration

1. Each Raspberry Pi should connect to the same LAN.

2. Assign static IP address to each Raspberry Pi for ssh remote control and management.

3. Using one computer to join this LAN for controlling the all Raspberry Pi.

4. Remote control other Raspberry Pi by ssh using *ssh hostname@ip address* with password.

3.3 Communication between different ROS hosts

Change the host file

Each host configuration file can be modify by *sudo vim /etc/hosts*. Add ip address and hostname in */etc/hosts* file for Raspberry Pi. Each ip address respond to one hostname which you give. Example hosts file is in the Appendix.

Running Talker and Listener

Enter ROS system in every Ubuntu system and ssh to com1. Then ping com1 and com2. If ping successfully meaning the network is fine, otherwise check the router and network problem.

The host name and ip address is respond to each Raspberry Pi, but the Master URI should be identical. Create a *.sh* file to export environment paths. See the appendix

Configure the file and execute this file in every Raspberry Pi host. Then run the talker and listener python file seperately in different hosts to check if listener can receive messages from talker.

Issues and Solution

Make sure setting the ROS hostname and IP address correctly and must explicitly set it. Otherwise the listener cannot receive talker because the host cannot find it in the network. The Talker will send messages continuously but listener can not receive these messages.

Check the hostname and ip address setting can use:

```
echo $ROS_HOSTNAME
echo $ROS_IP
```

Chapter 4

Experiment and Analysis

4.1 Experiment 1: Comparing different ROS1 version with previous experiment

Objective

This experiment investigates the performance in different frequency of message passing by using Wi-Fi network connection in a router. This experiment was executed by Issac in the previous and my aim is execute this experiment to compare it with the previous one[25].

Rationale

In order to acquire a detailed understanding of the performance characteristics of ROS communication channels, this experiment use two ROS hosts to commnunicate with each other and get the message latency results in different message frequency. I reproduced Issac's experiment to see what is the difference between my experiment and the previous one then analyze the reason to cause these differences.

Procedure

1. Start one master node on one host and send different frequency message to a echoer node on another host by using Wifi network.
2. Run the code which send timestamped messages from the sender host to the echoer host. Then the echoer host receive message and send it back to master host.
3. The master node will receive the message and then record message id and time in sending and receving into a file.
4. Message frequency is from 200 to 2000Hz (200,400,600...2000)which is same with Issac experiment.
5. Every frequency will be recorded 3 times with a range of message frequencies to obtain averaged results for each message frequency.
6. Running this experiment in different ROS1 version and operating system.

Hardware Configuration

4 Raspberry Pi 3B+
TP-Link 150M Router 

Software Configuration

Ubuntu 18.04 Bionic (64-bit)
ROS Melodic (Latest, Released May, 2018)
Ubuntu 16.04 Xenial (32-bit)
ROS Kinetic Kame (Released May, 2016)

Hypothesis

1. The message latency should be like the previous experiment results
2. Different message frequency may perform different latency. For example, lower message frequency can have lower message latency.

Results

The previous experiment from Issac is Figure 4.1. I only plot six different frequency instead of Issac 10 frequency because it may make the graph more clear to check out. The Figure 4.2 is message latency from ROS kinetic on Ubuntu Xenial and the figure 4.3 is ROS Melodic on Ubuntu Bionic.

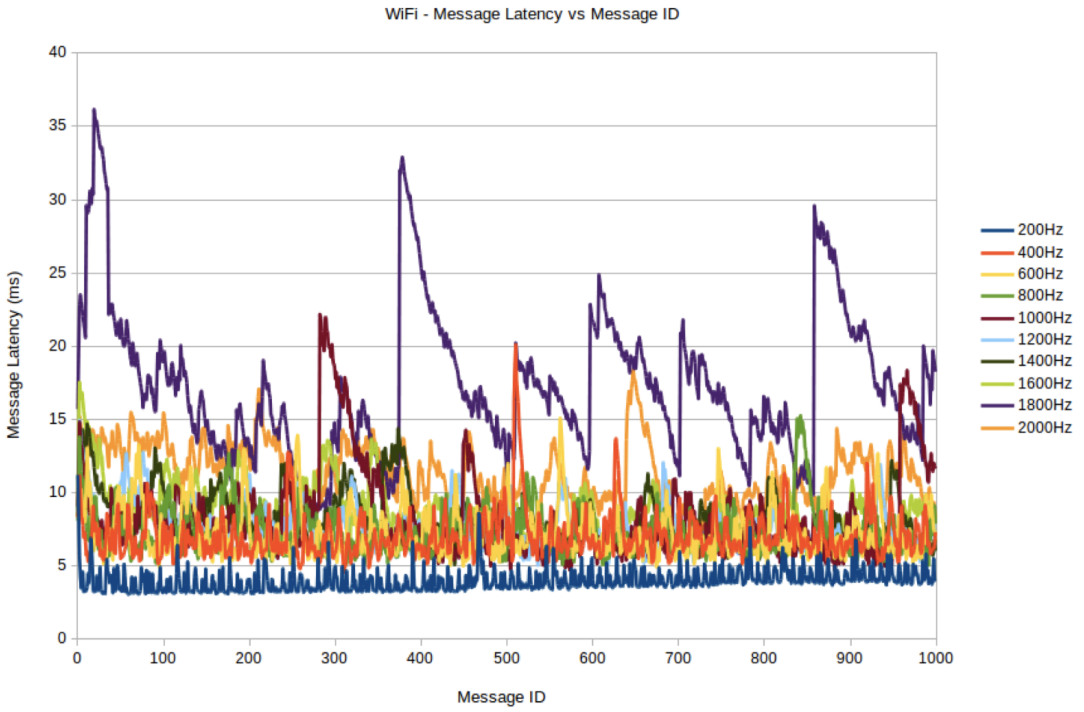

Figure 4.1: Message latency for all frequency in Issac experiment

Conclusion

1. The message latency from previous experiment by Issac is all almost under 20ms lower latency like 200Hz and 400Hz are very low latency and perform better than higher frequency like 1800Hz and 2000Hz.
2. In my experiment, message latency is high and very unstable in ROS kinetic on Ubuntu Xenial 32bits. Message latency can be up to hundreds millisecond which is ten times than previous experiment message latency.
3. Then in ROS Melodic on Ubuntu Bionic 64bits, the message latency is low and the result is very like the previous Issac's one. In lower frequency 200Hz and 400Hz show under 10ms latency and in higher message frequency, latency can be vary at 15-20Hz.
4. The ROS version and operating system version may have influence on the communication performance between different hosts. Router could also be a factor that have slightly impact on message latency

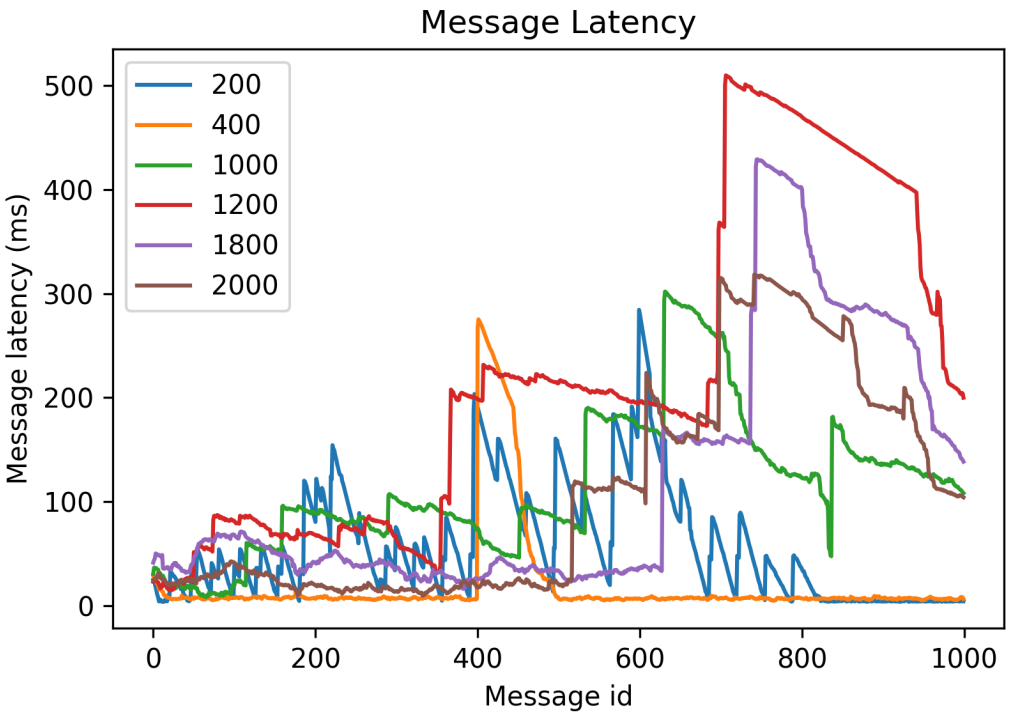

Figure 4.2: Message latency for ROS kinetic on Ubuntu Xenial 32-bit

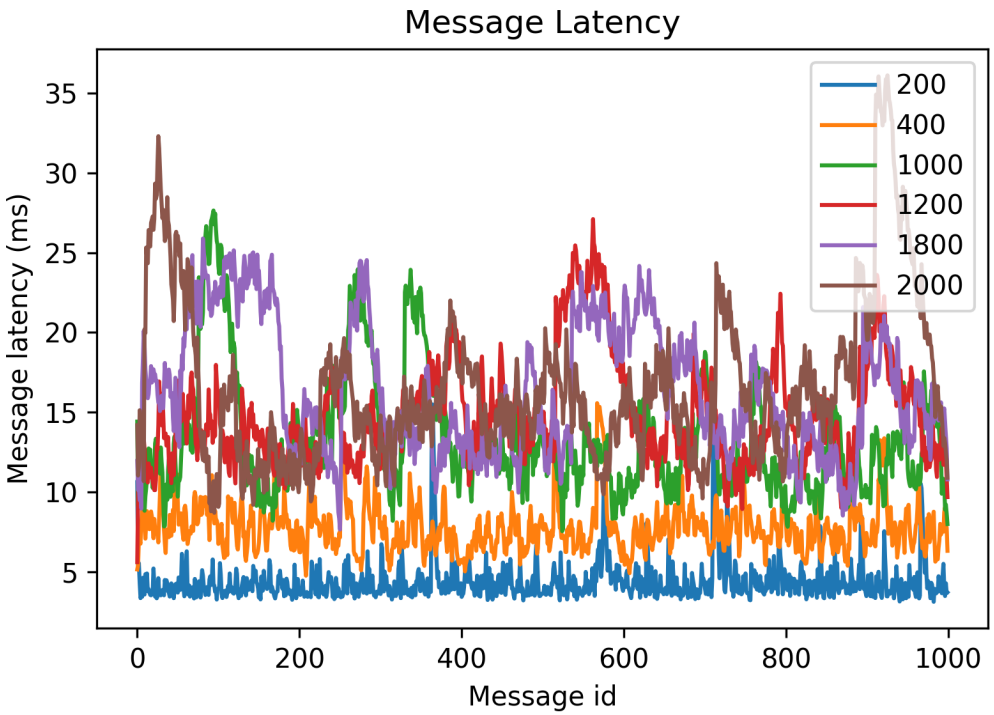

Figure 4.3: Message latency for ROS Melodic on Ubuntu Bionic 64-bit

4.2 Experiment 2: ROS1 and ROS2 message latency comparing

Objective
The aim for this experiment is to compare the message latency in communication between ROS1 and ROS2 in all other same condition. Then have a data analyze for different frequency message to conclude communication performance.

Rationale
In order to acquire a detailed performance characteristics of ROS1 and ROS2 channels, this experiment uses Raspberry Pi 3B+ which installed operating 64-bit ubuntu system. Then communicating by the same router with ROS1 and ROS2 separately. This can guaranteed the same variable and validity of this experiment.

Procedure
1. Start one node on one Raspberry Pi and send different frequency message to another host through wifi router.
2. Using ”rospy” in ROS1 run python2 code which send timestamped messages from the sender host to the echoer host. Then the echoer host send message back to sender host.
3. The sender node will receive the message and then record message id and time elapsed in communication into a file.
4. Message frequency is from low frequency (1Hz and 10Hz), middle frequency(100 and 200Hz) and high frequency (1000 and 2000Hz). Every frequency will be run 3 times and get a mean time results.
5. Using ”rclpy” in ROS2 run python3 code which also send and receive message and then record the results.
6. Record all result and plot by using python in Jupyter notebook.

Hardware Configuration
4 Raspberry Pi 3B+
TP-Link 150M Router

Software Configuration
Ubuntu 18.04 Bionic (64-bit)
ROS1 Melodic (Latest, Released May, 2018)
ROS2 Crystal Clemmys (Released December 2018)

Hypothesis
1. In ROS2 the message latency is lower than ROS1 overall.
2. High frequency has lower message latency than lower frequency in both ROS1 and ROS2.

Results
Figure 4.4 and 4.5 compare ROS1 an ROS2 message latency from 1Hz to 2000Hz message frequency. Figure 4.6 and Figure 4.7 compare 1Hz and 10Hz message frequency. Figure 4.8 and Figure 4.9 compare 100Hz and 200Hz message frequency. Figure 4.10 and Figure 4.11 compare 1000Hz and 2000Hz message frequency.

Conclusion
1. From the experiment, overall, ROS1 is less stable than ROS2 from low frequency to high frequency. Most message latency is under 50ms in ROS2, but massage can vary under 100ms in ROS1.
2. In low frequency, 1Hz and 10Hz comparing, ROS1 latency is not stable and sometimes shows

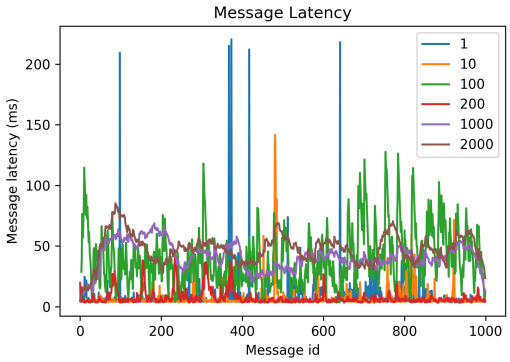

Figure 4.4: ROS1 all frequency

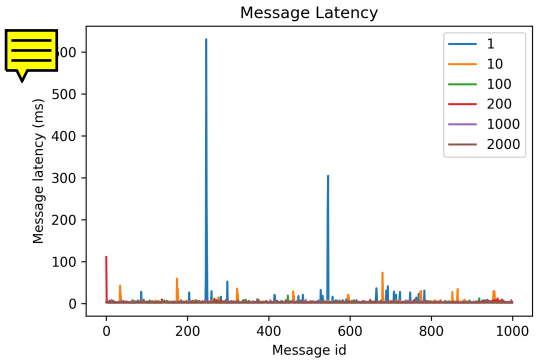

Figure 4.5: ROS2 all frequency

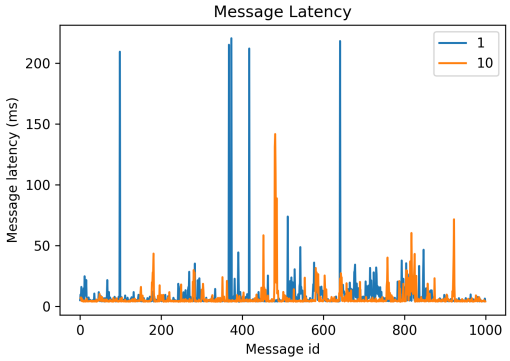

Figure 4.6: ROS1 1Hz and 10Hz

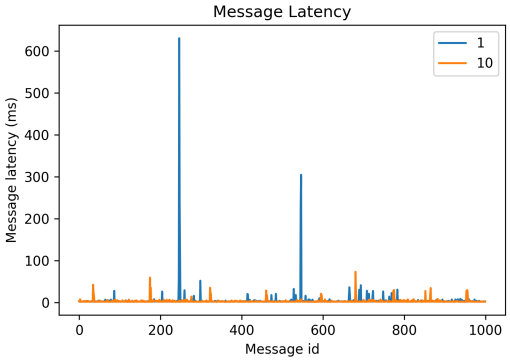

Figure 4.7: ROS2 1Hz and 10Hz

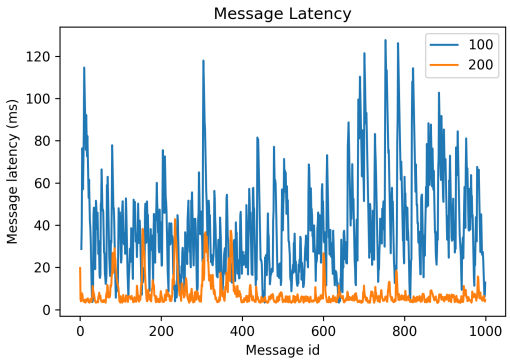

Figure 4.8: ROS1 100Hz and 200Hz

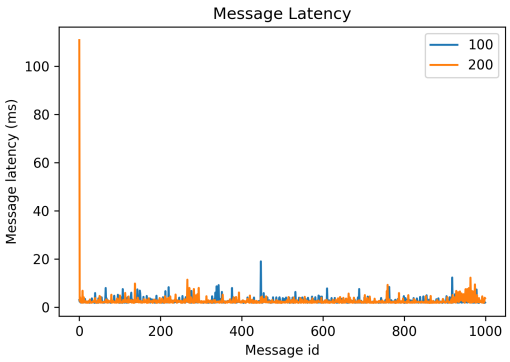

Figure 4.9: ROS2 100Hz and 200Hz

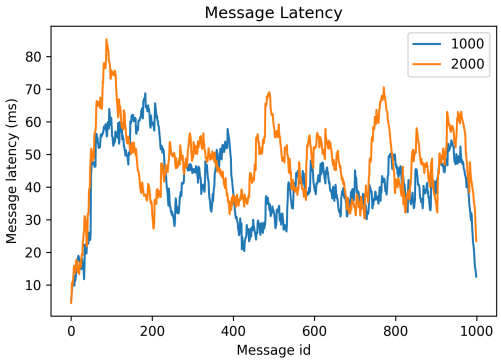

Figure 4.10: ROS1 1000Hz and 2000Hz

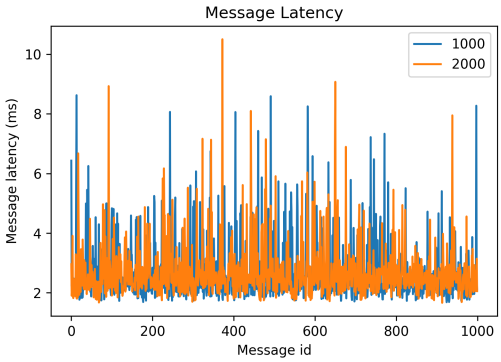

Figure 4.11: ROS2 1000Hz and 2000Hz

high frequency over 200ms. However, in ROS2, most message latency is very low, only few message shows high latency that might be the network problem.
3. Then in middle frequency, 100Hz and 200Hz message frequency, ROS1 message latency can vary from 20ms to 100ms. However, ROS2 shows a stable and low message latency under 10ms.
4. In High frequency, 1000Hz and 2000Hz, message latency is from 30ms to 70ms in ROS1, which is obviously higher than ROS2 under 10ms.
5. To conclude, ROS2 can show more stable and less message frequency than ROS1 with the same hardware and operating system. Although sometimes there will be network fluctuation causing very high message latency, it may be solved if using better hardware like router.

Chapter 5

Conclusion

5.1 Summary

5.2 Future work

Appendix A

First appendix

A.1 ROS1 installation commond steps

```
# Install system

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
sudo apt update
sudo apt install ros-melodic-ros-base

# Install tools and dependencies

sudo rosdep init
rosdep update
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt install python-rosinstall python-rosinstall-generator python-wstool
build-essential

# Start ROS service and create workspace

roscore
create catkin folder
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
```

A.2 ROS2 installation steps

```
# Install
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
sudo apt update && sudo apt install curl gnupg2 lsb-release
curl -s
https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key a
install
export CHOOSE_ROS_DISTRO=crystal
sudo apt update
sudo apt install ros-$CHOOSE_ROS_DISTRO-ros-base

# Environment setup
sudo apt install python3-argcomplete
source /opt/ros/$CHOOSE_ROS_DISTRO/setup.bash
echo "source /opt/ros/$CHOOSE_ROS_DISTRO/setup.bash" >> ~/.bashrc
sudo apt install python3-colcon-common-extensions

# create workspace
mkdir -p ~/ros2_example_ws/src
cd ~/ros2_example_ws
```

/etc/hosts

```
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

192.168.1.11 com1
192.168.1.12 com2
```

config.sh

```
# /bin/sh
export ROS_HOSTNAME=com1
export ROS_IP=192.168.1.11
export ROS_MASTER_URI=http://com1:11311/
```

Appendix B

Second appendix

Bibliography