CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY IN PRAGUE

RESEARCH REPORT

ISSN 1213-2365

Latency measurement of communication between DX100 controller and ROS

(Version 1.00)

Libor Wagner

wagnelib@cmp.felk.cvut.cz

CTU–CMP–0000–00

November 20, 2012

This work was supported by the EU FP7/2007–2013 (ICT 2011-2.1 Cognitive Systems and Robotics), under grant agreement no. 288553 (project CloPeMa).

Research Reports of CMP, Czech Technical University in Prague, No. 0, 0000

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Latency measurement of communication between DX100 controller and ROS

Libor Wagner

November 20, 2012

Abstract

This document describes a process of measurement of the Ethernet communication latency between Motoman DX100 controller and ROS node.

1 Introduction

The Cloth Perception and Manipulation (CloPeMa) project is concerned about robotic manipulation of cloth and other fabric. The robot used in this project is a industrial welding robot. It is controlled through customised firmware application using Robotic Operating System (ROS) from standard PC. Such configuration has many advantages over the standard industrial controllers, but also some disadvantages. One such disadvantage is the higher latency introduced by the network communication.

The latency consists of several components. The main component is the computation time needed for conversion of data-structure into data-stream and back. This time can be improved either by efficient computation or by assigning more resource to the process. The second component is the communication medium which in our case is Ethernet. These components are visualised in Figure 1.

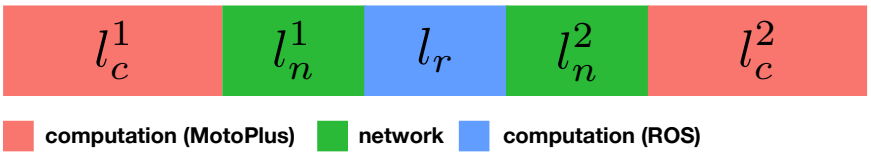

Figure 1: Latency components

These components are hard to measure separately. Instead, we measured the latency as a whole $l = l_c^1 + l_c^2 + l_n^1 + l_n^2 + l_r$. The l_n^1, l_n^2 and l_r are assumed to be constant, as the computation power of the PC is much greater than of the controller. On the other hand the computation time on the controller (l_c^1, l_c^2) can be influenced by number of factors. That include the priority assigned to the process, and number of other running task and extend of their utilisation.

2 Experiment description

The CloPeMa testbed consists of two MA1400 welding robots controlled by two DX100 controllers. The master controller is running VxWorks 5.5.1. with WIND 2.6. kernel. Along with our task the controller is running number of tasks that we are not able to control. These include crucial task such as motion control but also some useless such as ftp and http server. The controller is connected through HUB to local network.

The computer running ROS is standard Intel Core i7-2600 CPU 3.40 GHz, 8 GB of RAM, running Ubuntu 12.04 and the ROS Fuerte.

2.1 Implementation

The implementation of the experiment consists of two parts. First part is the controller task, implemented using MotoPlus library. It repeatedly sends message to the ROS task and waits for reply, while measuring time between the two. Second part is a ROS node which only receives wait for messages and send a reply.

2.2 Experiment parameters

In order to measure what is influencing the latency of the communication we have measured the latency in various configurations, generated by three parameters: application, priority and movement. For each configuration we have collected 1000 measurements.

Application parameter controls the number of other task running on the controller along the measuring task. We have tested two options. **Alone**, the task is running alone without other additional task (except that we can not control). **Included**, the task is running along with our standard application, that consists of three tasks; motion, position and io interfaces.

Priority parameter reflects the two possible priorities, that can be assigned to task using MotoPlus library. The priorities are **normal** and **critical**, that corresponds to priorities of 150 and 100 respectively.

Movement parameter describes whether the robot is moving and in what mode. The movement can be issued from **pendant** or from **ros**. Using pendant no additional task are required. However, when using ROS the whole application needs to be present (i.e. the test task must be included in the main application) in this case the position interface is streaming positions of the joints to the ROS and commanded joint coordinates are streamed to the controller. **None** represent the robot without motion.

parameter	**values**
application	alone, included
priority	normal, critical
movement	none, pendant, ros

Table 1: Summary of the parameters and possible values

3 Experimental results

The figure 2 shows that the biggest difference is between the experiments with low and high priority. This indicates that most of the time is spend by the computation. The type of movement has just a little impact in contrast to priority. Movement issued from ROS produced more diverged measurement with latency values up to 16 ms.

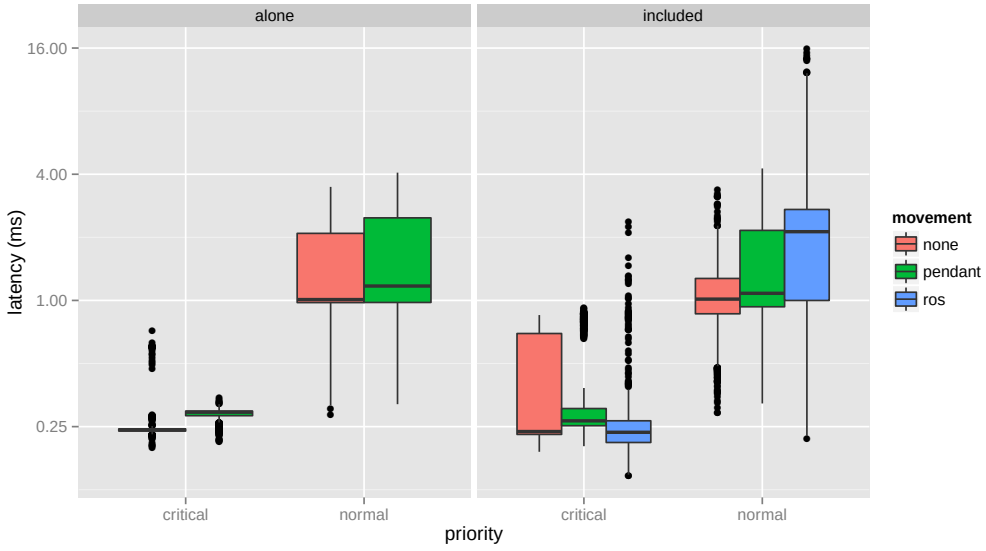

Figure 2: A boxplots[1] visulising all the measurements. The two plots shows measurement for task running alone and included in the application, X axis denotes the priority of the task, and color encodes the type of movement present during the measurement.

The figure 3 shows histogram of the experiment where the task was alone and without movement. The high prioritized task shows stable latency in the range from 0.2 to 0.4 ms. However, the latency of the low priority task is very unstable with values ranging form 0.2 up to 3.2 ms.

[1]The upper and lower hinges are first and third quartile. The upper whisker extends from hinge to the highest value that is withoun $1.5 * IQR$, where IQR is distance between the first and third quartiles. Similarly for lower whisker. See ggplot2 documentation for detailed description.

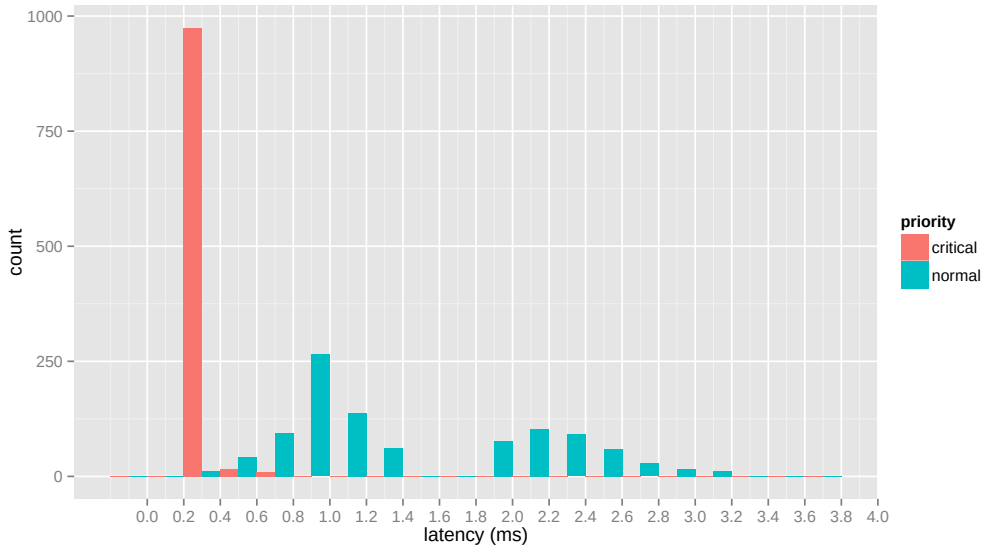

Figure 3: Histogram of the

4 Conclusion

The results shows that the priority have the greatest impact on the latency of the communication. The mean latency for the normal priority is 1.6 ms compared to the 0.3 ms of the critical priority. The movement has almost no impact on the critical task. When running with normal priority both pendant and ROS has