Evaluating the Scalability of ROS Multi-Robot Systems - Status Report

Isaac Jordan - 2080466

December 14, 2016

1 Project Description

This project is to create and undertake a set of experiments to identify the suitability, possible issues, and scalability limits of using ROS (Robot Operating System) in a multi-robot system. It first requires understanding of the simplest multi-host scenario (a pair of communicating nodes), then scaling up and analysing performance. The project would also involve the creation of a real-life demonstration of a multi-robot ROS system.

2 Progress

Several 'scoping' experiments have been completed to date. These experiments were designed to broadly track down areas of ROS worth investigating. At the beginning of the project there was already code by a prior student designed to measure message latency between two hosts running ROS nodes - however this code gave very unexpected results. The first task was to repeat this experiment using clean Raspberry Pi's with a fresh install of ROS, and verify that the results were the same as had been seen previously. This was successful.

The next task was to identify why these results were so different than expected - I reviewed the code for the experiment and identified plausible issues that could result in unusual message times. The cuplrit was a delay in the echoer (the host that echoes the message back to the original sender) - the echoer should always immediately re-send a received message instead of waiting for some time period to pass and then doing so. Rerunning the experiment after this fix gave the expected results - as message-sending frequency increased, message latency was the same until some frequency 'bottleneck' was reached, at which point message latencies sharply increased.

As a brief informal experiment, I modified the code to cache message times in memory (rather than write to disk during the experiment) in case the disk-writing was affecting processing times. However this modified version gave drastically worse times than the regular disk-writing, thus was abandoned. It was also noted at this time that the file writing mechanics of Python would result in in-memory caching of values anyway - thus this was not neccessary.

The next experiment was designed to validate the experimental procedure used in other experiments. It's aim was to investigate whether restarting the hosts running ROS between experimental runs (i.e. between message streams) affected performance. The rationale was that internal network buffers may fill up (and therefore slow performance), or cache messages (and therefore increase it) throughout a message stream. The experimental hypothesis by myself was that rebooting the hosts would make no difference to performance, however the results demonstrated a notable drop in performance (increased latency when restarting the machines often). Thus the decision was made that future experiments would have no system reboot between runs.

During previous runs it was noted that the Raspberry Pi's had high CPU usage on a single core during execution - one explored avenue was to split the sending and receiving functionality in to 2 seperate ROS

nodes (previously done in a single node). However when completed, the performance between the two versions (one node vs two node) was identical, and CPU usage was still high on a single core.

Thus the next experiment was designed to investigate how the CPU performance affected ROS message latency. The CPU in the Raspberry Pi was tested at 3 different speeds: default (1.2GHz), 75% (900MHz), and 50% (600MHz). The results from this experiment showed a direct correlation between A) the frequency at which messages began slowing down, and B) how slow the message times were overall. This shows that in the case of a small-payload, high frequency ROS message stream that CPU is the limiting factor, in the case of this specific system (ethernet-connected Raspberry Pi 3's).

Next was an experiment to investigate the difference in ROS message performance between using ethernet-connected hosts compared to WiFi-connected hosts. As in other experiments, a spectrum of frequencies were tested. The results were as expected, a wireless connection resulted in overall higher message latency, and also reduced the frequency at which messages started 'piling up' (taking longer and longer to deliver).

The last two experiments were to repeat the previous two (CPU and WiFi) using 'realistic data'. During previous experiments, the messages sent consisted of an integer message ID number, a timestamp, and a small constant string payload. However this message type does not represent a typical robotic middleware message. Thus it was decided to repeat the previous two experiments using A) sensor data, such as that from a Laser rangefinder and B) video data. These two data types represent two typical message sizes (4.25KB per message for sensor, and 308KB per message for video) that might be seen in a robot (data used is a subset of data recorded by a ROS robot at the MIT Stata Center, thus is actual real data).

A background (literature) review was also undertaken in this time period. This covers the background of robotics, multi-robot systems, scalability, robot middlewares, and then an overview of many of the various robot middlewares available, and a discussion of their design decisions. It also has a more in-depth discussion of how ROS is structured, and what hardware is being used throughout the project (such as the robot cars).

3 Plan

The next planned experiments will investigate the scalability of ROS in the horizontal (more host) direction, and vertical (each host doing more). This will require a series of experiments over the course of January (beginning week starting Jan 2nd, finishing by Jan 30th):

- Pair-wise Connected Experiments
  - Horizontal (2, 4, 6, 8 hosts)
  - Vertical (2, 4, 8, 16, 32, 64, 128, 256 nodes split on 2 hosts)
  - Both (2, 4, 8, 16, 32, 64, 128, 256 nodes split on 2, 4, 6, 8 hosts)
- Fully Connected Experiments
  - Horizontal (2, 4, 6, 8 hosts)
  - Vertical (2, 4, 8, 16, 32, 64, 128, 256 nodes split on 2 hosts)
  - Both (2, 4, 8, 16, 32, 64, 128, 256 nodes split on 2, 4, 6, 8 hosts)

The horizontal, and vertical experiments need not be conducted seperately from the 'both' experiment as they are a subset of that experiment. These experiments will be completely by the beginning of February.

Further, I plan on undertaking a demonstration project utilising as much existing libraries in order to save on development time. The project will utilise at least 2, but hopefully up to 9 of the robot cars

dedicated to this project, and will demonstrate the multi-robot capabilities of ROS (and hopefully some issues with it). The exact nature of this demo project hasn't been ascertained yet. This would begin immediately after the scaling experiments are completed (week starting Jan 30th), and would finish by the end of February (finished by Feb 27th).

The time period from the end of the demo project (Feb 27th), to the hand-in (Mar 21st) will be utilised primarily for writing the dissertation, and formatting and analysing results.

4 Problems

Problems experienced has mostly been due to installation of ROS on the Raspberry Pi platform. There are not pre-built binaries in the package manager, thus ROS must be compiled from source on the hosts - which can be troublesome due to low RAM resources. However these problems have been solved utilising prior knowledge of the Raspberry Pi systems, as well as online resources.

One possible future problem is an issue of time, as it is plausible that the scalability experiments will consume more than a month of work, thus running over in to the demo project's allotted time. This is due to the complex (and multidimensional) nature of the scalability experiments. There are many variables to modify to fully understand the interaction between them - such as number of hosts, nodes per host, communication topology, message format, and message frequency. In the case of run-over, the scalability experiments have precedence over the demo, as it is more useful to future research in this area. However, if the run-over is significant then the scalability experiments can be reduced in scope by only considering one communication topology (e.g. pairwise), or a reduced number of message frequencies and formats.