Remotely Visualizing and Controlling a Robot Swarm with ROS

MATTHEW LAWSON[1] AND GREGOR VON LASZEWSKI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: laszewski@gmail.com

project-000, May 2, 2017

Our project demonstrates the feasability of harnessing remotely-located distributed computing environments, i.e., "clouds", to simulate large-scale robot swarms. Our proof-of-concept program creates a two-robot swarm on a cluster of remotely-located computers. It then pushes a visual simulation of the robots to the remote user. Finally, it sends a single command to the robots in order to demonstrate the feasibility of networked communication with the robots. The project utilizes two software packages from the Open Source Robotics Foundation (OSRF). Namely, it uses the *Robot Operating System* to define, create and control the virtual robots. The OSRF's *Gazebo* simulation software provides visualization of the simulation. We use *cloudmesh*, *Ansible* and a *nix shell script to deploy the software to a distributed computing environment.

Keywords: Cloud, I524, ROS, Gazebo, Robot, Swarm

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IO-3010/report/report.pdf

INTRODUCTION

Simulating a single robot's actions and responses to its environment prior to real-world deployment mitigates risk and improves results at a relatively low cost. Therefore, it seems reasonable to conclude that simulating the actions and responses of a group of robots, e.g., a swarm, will also improve results at a low cost. However, deployment of an interconnected swarm of virtual robots on a remotely-located cluster of computers imposes additional requirements versus a locally-hosted single- or multi-robot deployment. For instance, accessing and configuring multiple computers presents a time and resource challenge in contrast to a single-host setup. In addition, network security measures, such as ssh keys and port access, impede ROS' intra-cluster communication capabilities. In order to address the unique requirements of a networked, remotely-located swarm, we create a multi-platform system to automate the creation and deployment of the virtual swarm.

VIRTUAL ROBOT SWARM COMPONENTS

Robot Operating System (ROS)

The Open Source Robotics Foundation's middleware product *Robot Operating System*, or ROS, provides a framework for writing operating systems for robots. ROS offers "a collection of tools, libraries, and conventions [meant to] simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms" [2]. The Open Source Robotics Foundation, hereinafter OSRF or the Foundation, attempts to meet the aforementioned objective by implementing ROS as a modular system. That is, ROS offers a core set of features, such as inter-process communication, that work with or without pre-existing, self-contained components for other tasks.

Figure 1 illustrates the ROS universe in three parts: a) the plumbing, ROS' communications infrastructure; b) the tools, such as ROS' visualization capabilities or its hardware drivers; and c) ROS' ecosystem, which represents ROS' core developers and maintainers, its contributors and its user base.

The modules or packages, which are analogous to packages in Linux repositories or libraries in other software distributions such as *R*, provide solutions for numerous robot-related challenges. General categories include a) drivers, such as sensor and actuator interfaces; b) platforms, for steering and image processing, etc.; c) algorithms, for task planning and obstacle avoidance; and, d) user interfaces, such as tele-operation and sensor data display. [3]

Gazebo

The Foundation also supports *Gazebo*, ROS' 3D virtual simulation software. "Gazebo...simulate[s] populations of robots in complex indoor and outdoor environments. [It] offers physics simulation at a much higher degree of fidelity [than gaming engines], a suite of sensors, and interfaces for both users and programs [4]." Gazebo's usefulness center on three main features: a) physics engines compatibility; b) its graphics engine;

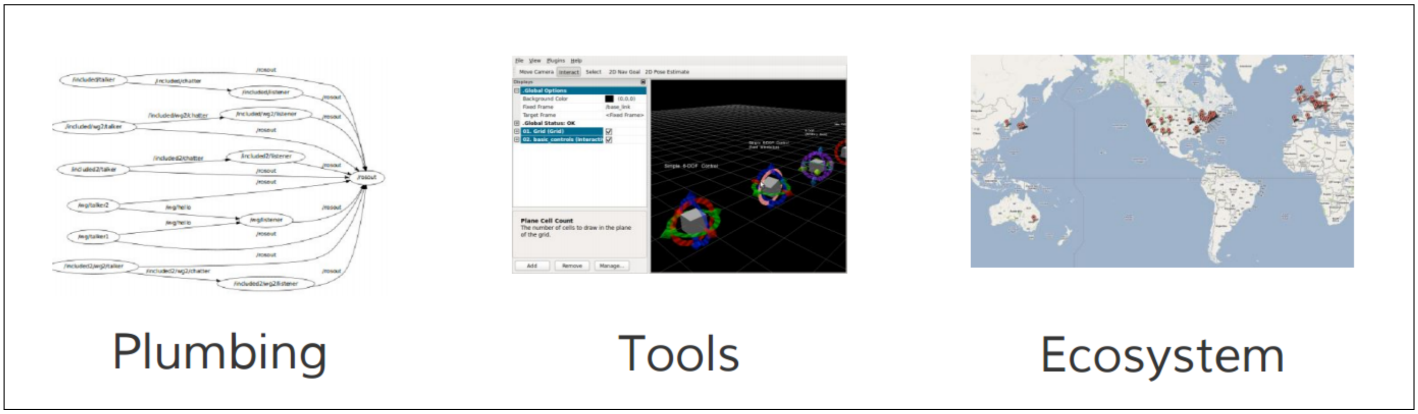

Fig. 1. A Conceptualization of What ROS, the *R*obot *O*perating *S*ystem, Offers to Roboticists [1]

and c) its sensor-data generators. with respect to physics engine compatibility Gazebo interfaces well with *Open Dynamics Engine* [5] (ODE), the default; b) *Bullet* [6]; *SimBody* [7]; and, *DART* [8]. Roboticists also benefit from its 3D graphics engine, *Object-oriented Graphics Rendering Engine* [9] (OGRE), which provides a C++ class library to "[abstract away] the details of using the underlying [graphics] system libraries like Direct3D and OpenGL [10]." Finally, Gazebo can supply *sensor* data to the virtual robot. Virtual sensor support ranges from 2D cameras to Kinect-style sensors. The system can also generate *noisy* data to better simulate real-world results.

Gazebo exists as a stand-alone project, suitable for use by programs other than ROS. However, it integrates tightly with ROS given its common ownership. In fact, the version supplied with a ROS installation automatically establishes communications between Gazebo and ROS for the end-user [11].

Ansible

Red Hat, Inc's [12] *Ansible* software purports to simplify numerous information technology tasks. It claims to do so by a) relying upon a human-readable script syntax, YAML; and b) by automating definable and repeatable IT tasks, such as configuration management and application deployment. Ansible's developers adopted a theater metaphor to describe the program's core functions. Thus, a computer's main duty within an IT infrastructure corresponds to the *role* an actor or actress might play in a theatrical production. Ansible calls the script a *playbook*, while the lines and directions within the script are referred to as *tasks*. Other aspects diverge from the metaphor, such as group vars and the config file (ansible.cfg). However, the *inventory* file hews to the metaphor - it represents the cast billing, the delineation of who plays what role. When used with an Ansible playbook, the inventory file specifies which servers belong to which logical group(s), i.e., which role(s).

As a result the software's applicability extends well beyond simplistic tasks even though Ansible's designers strive for simplification. In fact, an Ansible user can exercise fine-grained control over nearly every aspect of his or her IT infrastructure with a well-designed playbook.

Ansible also attempts to ease the burden of the IT administrator by eschewing SSL signing servers, daemons or client software. It simply pushes small programs to the target computers through an SSH connection to execute the desired tasks. When the task completes, Ansible removes the programs.

cloudmesh client toolkit

The *cloudmesh client toolkit* (cm) attempts to abstract away the complexities of establishing and utilizing different remotely-accessed computers and computer clusters [13]. Users can create, access and destroy a virtual machine or cluster of machines by issuing a single line of commands from a terminal emulator. cm supports access to clouds based on various back end-software stacks, including SLURM, SSH, Openstack and Heat. It provides an API, a command line client and a shell client.

Testing Environment

The Chameleon project's cluster of 650+ multi-core computer nodes, a joint venture between the University of Chicago and the Texas Advanced Computing Center, provides the infrastructre for the project. It is colloquially referred to as Chameleon Cloud or CC. A 100Gbps connection runs between the two centers. Project development primarily occurred on three-node clusters created as needed using Chameleon Cloud's *m1.medium flavor* of Ubuntu 16.04. The m1.medium flavor's resources consist of two virtual CPUs, 40GB of hard drive space and 4GB of RAM.

VIRTUAL ROBOT SWARM PROJECT IMPLEMENTATION

VR Swarm task

Although the swarm accomplishes the seemingly-trivial task of driving in circles, the real accomplishment rests in proving the feasability of automating the deployment of multiple controllable virtual robots on a remote cluster of computers and then visualizing them on a local computer.

Deployment

Achieving the aforementioned task requires coordinating a modestly-complex mix of shell commands, cloudmesh commands, Ansible commands and ROS / Gazebo commands. A shell script provides the automation for the shell, cloudmesh and ROS commands, while Ansible's *playbook* functionality handles the Ansible-focused portion. Only the initial step, retrieving and sourcing the shell script that begins the deployment process, requires user intervention.

***wget...begin* - Retrieve the Initalization Bash Script** Deployment begins when the user retrieves the initalization file, named *begin*, from the project's Github repository. The user then starts the execution sequence by sourcing the file with the command

```
> ./begin
```

from the directory where the *begin* bash script resides.

```
++++Local Computer+++++
| > ./begin           |
| -sets up cluster    |
| -captures cluster ip|
|   addresses         |
| -places ROS robot   |
|   files on main node|
| -places Ansible     |
|   files on main node|
+++++++++++++++++++++++
 |        ^
 |        |
 |        ^
 |        |
 |   ++++++++Git Repo+++++++
 |   | Source for:         |
 |   | -wget...begin script|
 |   | -ROS robot files    |
 |   | -Ansible files      |
 |   | -beginAgain script  |
 |   +++++++++++++++++++++++
 |
 |     ++++++++Main Node+++++++
 |     | > ./beginAgain       |
 |     | -runs Ansible script |
 |     |  -installs catkin    |
 |     |  -installs ROS       |
 |-->  |  -modifies /etc/hosts|
       |  -sets up ROS env,   |
       | -builds robot pkg    |
       | -initalizes robot sim|
       | -sends commands to   |
       |   simulated robots   |
       ++++++++++++++++++++++++
             |           |
             |           |
++++++++Node #2+++++++   |
| -installed software|   |
|    -catkin         |   |
|    -ROS            |   |
| -listener.py       |   |
++++++++++++++++++++++   |
                         |
    ++++++++Node #3+++++++
    | -installed software|
    |    -catkin         |
    |    -ROS            |
    | -talker.py         |
    ++++++++++++++++++++++
```

Fig. 2. Deployment Workflow for the Virtual Robot Swarm Project

***begin* bash script** The bash script calls three cloudmesh_client commands, > cm cluster define −n rosA1 −c 3 , > cm cluster allocate and > cm cluster cross_ssh in order to create and prepare the three-node cluster named *rosA1*. The script then uses two additional Cloudmesh commands, > cm cluster nodes and > cm vm ip show to capture the public and private ip addresses of the cluster.

Modifications, Pitfalls

TBD; discuss any obstacles encountered with deployment due to dependency problems, connecting ROS and Gazebo, etc.

Initializing the Swarm

TBD; starting ROS and Gazebo to create the virtual environment; testing swarm interconnectivity; designating master node, etc.

Begin Task and Monitor Swarm's Progress

TBD; discuss the steps to initiate task completion and monitor the swarm's progress;

Information Acquired

TBD; discuss the iformation obtained from the swarm wrt the task at hand as well as each node's vital signs, e.g., battery level;

VR SWARM PROJECT CONCLUSIONS

TBD; present the data collected in some visualization format; discuss why this project advances robotics forward by utilizing distributed computing;

SUPPLEMENTAL MATERIAL

REFERENCES

[1] H. Boyer, "Open Source Robotics Foundation And The Robotics Fast Track," web page, nov 2015, accessed 19-mar-2017. [Online]. Available: https://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/11/rft-boyer.pdf

[2] Open Source Robotics Foundation, "About ROS," Web page, mar 2017, accessed 16-mar-2017. [Online]. Available: http://www.ros.org/about-ros/

[3] National Instruments, "A Layered Approach to Designing Robot Software," Web page, mar 2017, accessed 18-mar-2017. [Online]. Available: http://www.ni.com/white-paper/13929/en/

[4] Open Source Robotics Foundation, "Beginner: Overview: What is Gazebo?" Web page, apr 2017, accessed 30-apr-2017. [Online]. Available: http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1

[5] Open Dynamics Engine, "Open Dynamics Engine Wiki," Web page, feb 2017, accessed 30-apr-2017. [Online]. Available: https://www.ode-wiki.org/wiki/index.php?title=Main_Page

[6] Real-Time Physics Simulation, "Bullet Physics Library: Real-Time Physic Simulation," Web page, apr 2017, accessed 30-apr-2017. [Online]. Available: http://bulletphysics.org/wordpress/

[7] P. E. Michael Sherman, "Simbody: Multibody Physics API," Web page, apr 2017, accessed 30-apr-2017. [Online]. Available: https://simtk.org/projects/simbody/

[8] Georgia Tech and Carnegie Mellon University, "DART: Dynamic Animation and Robotics Toolkit," Web page, mar 2017, accessed 30-apr-2017. [Online]. Available: http://dartsim.github.io

[9] Torus Knot Software Ltd, "OGRE3D," Web page, apr 2017, accessed 30-apr-2017. [Online]. Available: http://www.ogre3d.org

[10] Torus Knot Software Ltd, "OGRE: About," Web page, apr 2017, accessed 30-apr-2017. [Online]. Available: http://www.ogre3d.org/about

[11] Open Source Robotics Foundation, "gazebo-ros-pkgs: Package Summary," Web page, aug 2016, accessed 30-apr-2017. [Online]. Available: http://wiki.ros.org/gazebo-ros-pkgs

[12] Red Hat, Inc., "redhat," Web page, apr 2017, accessed 30-04-2017. [Online]. Available: https://www.redhat.com/en

[13] G. Laszewski, von, "Cloudmesh Client Toolkit," Web page, apr 2017, accessed 30-apr-2017. [Online]. Available: http://cloudmesh.github.io/client/

AUTHOR BIOGRAPHIES

Matthew Lawson received his BSBA, Finance in 1999 from the University of Tennessee, Knoxville. His research interests include data analysis, visualization and behavioral finance.

WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Matthew Lawson. Designed the project in collaboration w/ Gregor von Laszewski, researched the material and implemented the project. Slept far too little.

Gregor von Laszewski. Provided invaluable insights at key points during the process.