RoboCup Rescue Simulation Competition Manual

RoboCup Rescue Simulation Team

Version 1.1, February 06, 2023

Table of Contents

1. Purpose

The manual describes the sequence of tasks to setup and run the RoboCup Rescue Simulation competition on a cluster of Linux computers.

2. General Notes

This manual assumes the RoboCup Rescue Simulation competition will run in a cluster of Linux machines.

2.1. Software Requirements

- OpenSSH
- Java OpenJDK 17
- Git
- Utilities like bash`, `xterm`, `tar`, `7zip`, `gzip`, etc.
  Note: If you are using Ubuntu, all of these utilities are present in the default software repositories.

3. Setup

The competition environment is composed of 5 Linux computers, one controller called `controller` and four simulation computer called `c1-1`, `c1-2`, `c1-3`, and `c1-4`. The `c1-1` runs the RoboCup Rescue Simulator, while each other computer `c1-2`, `c1-3`, and `c1-4` runs a specific type of agents (i.e., ambulance teams, fire brigades, and police forces).

3.1. Installation

In Ubuntu, the installation proceeds according to the following commands.

Installation on Ubuntu

```
$ git clone https://github.com/roborescue/rcrs-server.git
$ cd rcrs-server
$ ./gradlew completeBuild
```

You can download the simulation server by cloning the `https://github.com/roborescue/rcrs-server` repository. Clone the simulator server using the command

```
git clone https://github.com/roborescue/rcrs-server.git
```

General Notes: Getting server access for the first time: ssh-keygen ssh-copy-id

rescue@116.203.191.59

Login to the server: ssh -X rescue@116.203.191.59 Validate display is connected by typing in the terminal: “echo $DISPLAY” or “xcalc” check.sh ⟶ checks cluster connection

To access the clusters: Cluster 1: ssh c1-1 Cluster 2: ssh c2-1

Teams names in the scripts: AIT, CSU, MRL, RI1

Preparing for competition day: nano scripts/remote-control/config.sh ⟶ to select the right day, ex: DAY = Day1, DAY = Final

Copy maps to clusters: Check map exists in /home/rescue/in_maps/submission_day/ Copy map to /home/rescue/maps/ ⟶ cp -r paris/ ../../maps/paris3 ./copyToServers.sh maps/ ⟶ maps folder should contains maps directly, ex: maps/paris3

Check teams submitted their code: Teams codes should be in /home/team_name/code/submission_day_time/

Copy new submitted code to clusters: cd home/rescue/codeDir rm -rf in/* rm -rf out/* ./fetchCodes.sh ⟶ nano before copying & validate the code version is correct, ex: 10 = day 1 first submission, 11 = day 1 2nd submission….etc ./prepare.sh teams_name (ex: RI1, MRL) ./uploadCodes.sh cd .. ⟶ go back to rescue folder ./copyToClients.sh code/

Run maps: ./run.sh [cluster_number] [map] [team_name] Example: ./run.sh 1 berlin1 AIT ./cancelRun.sh (1 or 2) ⟶ if there are any errors

Generate logs: nano scripts/evaluation/config.py ⟶ make sure teams, maps per day & days are added properly evalAll.sh ⟶ if this doesn’t work then run evalLog for each team evalLog.sh [path_to_gz_file] [map] [team_name] Example: evalLog.sh logs/Day1/kernel/0624-104636-CSU-Yunlu-eindhoven1.gz eindhoven1 CSU

4. Notes

evaluation = log execution code = team code codeDir = prepare team code codeDir/in = team code

/fetch Codes to fix copy

nn = variable

/copyToServers.sh maps/

- maps with the proper numbering

4.1. Run the competition

4.2. Update Web Interface

/evalAll.sh

scripts/remote_control/config.sh scripts/evaluation/config.py

logs/Day1/kernel = Log of teams