Remote Observing at CHARA
CHARA Technical Report No. 102

Jeremy Jones

July 24 2020

1 Introduction

The CHARA Array's current remote observing system has been in operation since June 2018. In that time, it has been used for over 200 nights ($> 30\%$) of observing time (pre-COVID-19). The remote observing system has been used to observe with all currently active beam combiners and for both internal programs and open access programs (through NOAO/NSF NOIRLabs). This report describes the remote observing system, how it is used, and how it is maintained. In Section 2, I describe the remote observing system and the software installed on it; in Section 3, I explain how the system is accessed and used for remote observing; and in Section 4, I discuss how the system is maintained.

2 Description

The primary component of the remote observing system is a virtual machine (VM) running the Ubuntu 16.04.4 Operating System installed on the CHARA Server in the Georgia State University Data Center in Atlanta. All necessary software for conducting remote observations have been installed on this VM. In most cases, this is exactly the same software that is used on site at Mt. Wilson, but in a few cases, the software has been optimized for the remote user's experience. The software interfaces, running on the VM, connect to their respective servers, running on Mt. Wilson, using SSH port forwarding (often referred to as SSH tunneling). The remote user accesses the VM using a Virtual Network Computing (VNC) connection. This setup allows the user to have a fast and reliable connection between Atlanta and Mt. Wilson despite limited bandwidth on the mountain while the more bandwidth-heavy graphical connection takes place between the user and Atlanta.

2.1 Installed Software

The CHARA software packages[1] that are installed on the remote observing VM are listed in Table 1. In addition to these software packages and their prerequisites, we have installed the following:

- The VEGA control software

[1]These packages can be found at https://gitlab.chara.gsu.edu/.

- rdesktop for connecting to the VEGA windows machines on the mountain
- logwatch, Fail2Ban, and Denyhosts for security and system monitoring
- TigerVNC for running our VNC servers

Table 1: Installed CHARA Software Packages from the CHARA Gitlab Server.

Software Package	Description
central_scrutinizer	Main sequencer for the CHARA Array
chara-array-software-libraries	Source code for all libraries used on the CHARA Array control system
chara-database-utils	Tools for maintaining the CHARA object database
chara_scripts	Bash scripts used for CHARA Array control
chara-standalone-camera-server	Tools for a stand-alone camera server utility
climb	CLIMB beam combiner control software
dome	Server and GTK interface for dome server of the CHARA Array
etc	Etc directory for CHARA control system
gps	GPS control software
hut	Server and client for controlling the huT hardware of CHARA telescopes
jouflu	Jouflu Control software
labao	Software for controlling the LABAO system
metrology-related	Repository for metrology-related software
mircx	MIRC-X control software
motion_control	CHARA Array control code for all servers and clients that move things
obsgtk	Integrated GTK GUI used to control telescope, finder, acquisition, etc.
ople	Server and GTK client code for OPLE control software
pavo	PAVO control software
refcam	Reference camera control
remote-power	Code to enable use of the Baytech RPC (remote power control) units
remote_obs_scripts	Scripts used for remote observing
socket-manager	Server and client code to run the CHARA Array socket manager
telescope	Telescope control software
tiptilt	All code required to run CCD based RT-Linux Tiptilt system
vacuum-monitor	Code to control the unit which monitors vacuum gauges
weather-monitor	Code to control weather monitoring units
wfs2	Wavefront sensor control code

Lurking mode is available (and enabled by default on the desktop icons and applications menu items) for the telescope GTKs and the weather GTK. In both cases, it can be enabled with the '−L' flag if using the command line. In the case of the telescope GTK, lurking mode reduces the number of buttons and tabs available to the user, and in the process, decreases the size of the window. In the case of the weather GTK, lurking mode eliminates the final two columns of the display (Ttel and RHtel). These two columns require communication with the HuT servers, which is not allowed remotely and, if not eliminated, delays the opening of the GTK for several minutes while attempting to access the HuT servers.

Within the obsgtk software package is a piece of software called lurkgtk, which provides displays for the telescope and lab adaptive optics (AO) systems as well as information about the status of the telescopes and their associated telescope and lab AO systems. Because obsgtk does not work as expected on the remote system, users are recommended to use lurgtk instead for monitoring the status of the AO systems.

2.2 Remote Observing Scripts

There are several scripts that are used exclusively on the remote observing VM: check_gitlab_updates, check_ssh_tunnel, download_weather_logs, kill_all_ssh_tunnels, kill_remote_ssh, and telescope_launcher. These are located in the "Remote Observing Scripts" gitlab repository[2]. All except the kill_remote_ssh script, which is a shell script, are written in python (version 3). When installed, all six scripts are copied to the /usr/local/bin directory without file extensions (e.g., check_ssh_tunnel.py is copied as /usr/local/bin/check_ssh_tunnel).

These six scripts do the following tasks:

- check_gitlab_updates checks a list of gitlab projects to see if they have been updated in the last 24 hours. If they have, the script sends an email to the Data Scientist. This is run daily using cron.
- check_ssh_tunnel checks if the SSH tunnel is running. If it is, the user has the option to end it. If it is not, the user has the option to start it. A desktop icon and applications menu item have been created to run this script and users of the remote observing machine should do this at the beginning of each observing night.
- download_weather_logs downloads the weather logs from /ctrscrut/chara/data so that the user can see the plots generated by the weather GTK. This script is run every five minutes using cron.
- kill_all_ssh_tunnels kills all active SSH tunnels. This script is run twice daily using cron as the root user - once well before observing begins (specifically, at 3:50 PM pacific time) and once well after observing ends (12:00 PM pacific time). This ensures that redundant SSH tunnel commands do not clog the system.
- kill_remote_ssh kills the oldest SSH tunnel that is open. This script is used by check_ssh_tunnel and kill_all_ssh_tunnels
- telescope_launcher launches a GUI where the user can select any or all telescope GTKs to open in lurking mode. A desktop icon and menu item have been created to run this script.

2.3 User Accounts

Primarily, remote observations will occur using the vroc account. However, accounts have been set up for the Data Scientist, the Director, and the MIRC and VEGA teams for ongoing development of the system. In addition, the aforementioned indiviudals and groups, as well as the Associate Director and Night Operations Manager have dedicated VNC servers available for their use.

[2]https://gitlab.chara.gsu.edu/jeremyjones/remote-obs-scripts

3 Access

The remote observing VM is not publicly accessible via the SSH protocol[3]. Our users connect to it via VNC. Five days[4] before the start of a remote observing run, the CHARA Data Scientist will start a VNC server that is to be used for the run, and will send the login information to the PI for the run. These login credentials will take the remote observer to a virtual desktop for the vroc account. Once logged on, the remote observer can check whether the SSH tunnel is active and otherwise, set up for observing much like they would on the mountain.

Before the COVID-19 pandemic, only users who had been trained on the mountain would be given access to the remote observing system. As a result of the pandemic, we have moved to completely remote observing with no site visits for observers. With these new protocols in place, we allow new observers to observe remotely only with the assistance of experienced observers.

4 Maintenance

The following is a list of procedures that are done to maintain the remote observing system:

- Check the observing schedule daily:
  - Two weeks in advance (give or take a weekend) of an observing run, inform the PI that they have observing time coming up and that they should fill out the remote observing sign-up form.
  - Five days in advance (give or take a weekend) of an observing run, set up a VNC server for the observer:
    - Check the remote observing machine status google spreadsheet to see what VNC servers are in use.
    - Double check that the server is operational by connecting to it with a VNC viewer.
    - If necessary, exit out of any open windows on this VNC server
    - Assign it to the PI of the observing run by noting it on the spreadsheet and emailing the PI the server address and current passwords.
- Check the server logs daily (using logwatch) watching for the following:
  - Unauthorized login attempts
  - Suspicious sudo commands
  - How much hard drive space is available
- Update the observing software weekly or as needed.
- Run system updates monthly.
- Change the VNC password at the beginning of the observing season or as needed.

[3]A user can only access the VM via SSH if they are on GSU's internal network.
[4]Or the Monday thereafter if this falls on the weekend.