ABOVE SERVER SOFTWARE DOCUMENTATION V 1.0

November 6, 2014.

Casey Daniel

This is the documentation for the ABOVE Server software. There are several parts to this software package, including TCP servers and the test client, the ingestion software that includes the file manager and Matlab plotting scripts, and the shell scripts. These are divided into appropriate subdirectories.

/tcp\_server

General Use

To start and stop these servers, use the following commands:

* ./PATH/TO/DIR/tcp\_server/tcp\_server\_PORT\_NO.py start
* ./PATH/TO/DIR/tcp\_server/tcp\_server\_PORT\_NO.py stop
* ./PATH/TO/DIR/tcp\_server/tcp\_server\_PORT\_NO.py restart

These execute the start, stop, and restart routines. For example to start the port 25000 server, in the directory as of the time of this writing, use ./data/vlf/src/tcp\_server/tcp\_server\_25000.py start.

There is also another command that is used for debugging purposes. In place of start, stop, or restart ‘main’ can be used. This will not demonize the script, and the process will be attached to the terminal session. This should only be used for debugging purposes.

If another server port is needed, simply copy and paste the entire server into a new python file, and change the port number variable to the one desired.

General Flow

While I do plan to create a flow diagram, I will write one out anyways. On start up the server will dethatch the process from the terminal. This ensures that closing the terminal session will not quit the program. The logging file is initialized as well as the TCP server on the assigned port. Once this is completed the server sits in an infinite loop looking for connections.

Once a connection is established a new thread is started, and waits for the control wakeup string. Once this is found a reply of control awake is sent, and another function is called to handle the receiving of the header string, followed by the data. This processes involves the header file being sent, the server checks the number of entries, as well as some values. If all is well the server replies with header ok, and the data will now be sent. Otherwise a string of header is not ok is sent to the instrument, and will attempt no more than 3 resends. After 3 attempts of receiving data in a connection the connection will then be aborted and closed.

Once a data file has been transferred, the file is then written to the root/rawData directory, with a file name structure of YYYYMMDD\_HHMMSS\_SITE\_##.chunk.dat.

The IP dictionary containing the list of IPs is also updated if needed and saved.

/Ingestion\_software

This section contains a number of different scripts for ingestion of the data. These files handle the data stream, and will contain other scripts to process them further as the project develops.

File\_manager.py

 This python script takes the data chunks from the raw data directory, combines the files, and updates header information for this file, and the full file and the chunks in their respective trees. Currently the full files can be found under root/fullFiles/YEAR/MONTH/DAY/SITE/HOUR, where hour is in UT, the chunks are the same tree of root/chunks/YEAR/MONTH/DAY/SITE/HOUR. This script is also responsible for information being sent to RTEMP (Real Time Environment Monitoring Platform). Various data is sent to them. If you wish to change the data sent, all that is required is the data string be updated to reflect the key, followed by the information. Packets are then sent via UDP to the server, and displayed on above.rtemp.ca.

 To start this script, use the same method as the TCP server.

plot\_maker.m/full\_plot\_maker.m

These two scripts produce the spectrogram summary plots. A simple FFT filter is applied (found in FFT\_filter.m) to integer multiple frequencies of 60Hz. This filter searches for the peak around these frequencies, and removes the selected amount of bins surrounding the peak. This is all to suppress the noise. This filter should be tested to ensure it actually does filter. These scripts use the full data files, found in the appropriate directory. They are then placed in /root/summaryPlots/YEAR/MONTH/DAY/SITE/HOUR directory. The difference in these directories is full\_plot\_maker.m searches for full files in the entire directory tree, plot\_maker.m only searches the previous and current day’s directory.

availability\_plotter.m/full\_availability\_plotter.m

These Matlab scripts are responsible for the creating the availability charts. This process is done, for each month, checking the hour directories for each site to see if they exist. This is done in 6 hour chunks. If all 6 are found, the appropriate value in a matrix is set to 2, if some are found the value is set to 1, otherwise the value remains 0. This matrix is then plotted. The difference for these to scripts is full\_availability\_plotter.m will look through all years listed in the “Years” vector, while availability\_plotter.m only searches this month.

This script needs to be manually adjusted when new sites are added. This is simply done by adding the string for the 4 letter site name. This needs to be done for both site Years also need to be update, again this is done by adding a year into the year vector and is only required in the full\_availability\_plotter.m.

Print\_ip.py

This is a simple python file that will print the IP dictionary saved by the TCP server, and read by the file manager and prints them to the screen. It may be useful to ensure the directory information is up to date in the file first.

/SummaryPlotViews

This directory contains the html page used on the above webpage for viewing the summary plots. This is a simple html page that contains a JavaScript. Utilizing a simple form the user can browse summary plots and availability charts. Using an ftp server hosted on the server, the image is then requested and loaded.

A few notes for this script. The year field must be manually updated every year. To do this the HTML file must be updated, then through drupal uploaded. Click on the file to get the file name and directory. This needs to be inserted under content>blocks. The block for the summary plot viewer has an html button, and a simple javascript to open the tool in a new page. The URL must be updated in this java script when the page gets updated.

/shellScripts

These are simple shells scripts that are called by various cronjob. The Matlab scripts are responsible for the plotting of the summary plots and availability charts. Reboot.sh is called on boot to start the FTP servers, and root.sh is called on boot by root to start the ftp server.

FTP server

Currently VSFTPD is used for the FTP server, anonymous is allowed and /data/vlf/summaryPlots is the current root directory. Nothing to fancy here.

Daemon

All the python scripts are demonized using code found on gitHub, and used with permission. Please see <https://github.com/serverdensity/python-daemon> for more details on use and documentation.