100505349 Distributed Systems Usage Document

*100505349*

Contents

[Setup 2](#_Toc89537693)

[Commands & Usage 3](#_Toc89537694)

[Load Balancing 4](#_Toc89537695)

# Setup

To begin using the program, there are some measures that must be taken to permit the system to work as expected. First and foremost, there are two documents that must be edited and coordinated, the \_ConnectionList.txt and the \_Logins.txt files. The connections list file must contain the list of all IPs that the server will run upon, delimited by a single “|” character, as shown below.

The login file must then be appended with similar data, storing the usernames and passwords of all valid logins to be processed by the device. It is assumed that this document will contain no duplicate usernames, and will be formatted in a system of “*Username*|*Password*”, followed by a new line, as shown below.

As previously mentioned, these files are expected to contain the same data for all instances of the system, and any modifications not mirrored on separate devices may cause unexpected issues to arise. This restriction is also true for any code files, which naturally should not be edited in any form. Outside of this, the contents “Music” folder and “cache” folder can be modified, notably, when attempting to test the distribution of files, different WAV files can be added to the “Music” as appropriate.

After performing this preliminary setup, the system can now be started, this can be done by starting the “Node.py” file, afterwhich the user is prompted to enter the name of a node to start. The only valid inputs for this are “Control” and “Client”. The “Control” node should be started first, and only started on a device that is listed in the \_ConnectionList.txt file – as many control nodes can be started as desired, though the system requires atleast one node and only permits one control node per device. A Client node can be started on any device on the same local network as the other control nodes, and is not restricted to the IPs referenced in the \_ConnectionList.txt file.

When run for the first time, there is a small chance the script may instantly close, this is due to circumstances outside the control of the program, in these rare cases the program may simply be started again to successfully run. An additional article of note is that the program has three restrictions in place – the system must have access to the ports 50001 through 50010, the system must have an active internet connection (Though this is not used for any functions aside from returning the IP the system should listen upon on the local device) and the system must be running on the windows 10 operating system.

Finally, due to restrictions set in place by the university, there is a chance that a node running on port 50011 or higher can produce anomalous results. As the system functions by checking the next available port after 50001, it is possible to reach this port number by having too many instances of the node file running simultaneously. This issue is not apparent in systems without the 50001-50010 port restrictions in place, but nonetheless a warning is displayed when a node reaches the 50011th port.

# Commands & Usage

When atleast one control node exists on an IP specified in the \_ConnectionList.txt file, and atleast one client has started, the client may input a number of commands into the system to achieve different results. A list of valid commands is provided below, with any user specified inputs being defined in brackets.

**Authentication Commands**

LOGIN|(Username)|(Password)

This command allows a client to submit a username and password to be processed by an authentication node, which will provide the user with the ability to access other commands if the username and password specified are valid. This command must be used before all other commands can be processed

**Music Download Commands**

!PLAY:(Music name)

This command, when provided with a valid music file name, will start the download of a music file from a file distribution node, allowing a client node to download music data across the network. This system has been tested with up to 15 minutes of music without any faults occurring, though this relies on the strength of the local network

PLAYLIST

This command returns a list of all available files on the network and is handled by the control node. Due to the inner workings of the network, the playlist will sometimes take time to update as new nodes connect, especially if there are multiple IPs in the \_ConnectionList.txt file without active control nodes

**Echo Commands**

ECHO|(String to echo)

This command returns a string back to the user with the same data as they initially input and is handled entirely by the echo node

ECHODUMP

This command, once again handled by the echo node, displays the values of all echo requests that have been processed on the node in question. This data is node specific, meaning it only shares the echo data provided by clients connected to this specific instance of the echo node.

**Dictionary Commands**

DICT|(String to define)

This command returns the definition of a value as stored in the internal dictionary of the handling dictionary node. Much like the echo node, the contents of this dictionary are node specific.

DICTADD|(Dictionary entry name)|(Dictionary entry description)

This command allows the input of a dictionary entry into a dictionary node, and much like the previous commands is node specific. Dictionary entries added can have their descriptions returned by the “DICT” command

**Playback Commands**

VOLUME|(Integer volume between 0 and 100)

This command, processed by the client itself, sets the playback volume to a percentage value. This operation can be completed while no music is playing and like other playback commands requires no authentication to be used.

PAUSE

This command, once again processed by the client node, stops the playback of a song temporarily.

RESUME

This command resumes the playback of a song that has been stopped using the “PAUSE” command

STOP

This command stops the playback of the song and discards the song data, meaning the song cannot be resumed and must be redownloaded if required.

These commands can only be requested by the client, as the server nodes have no functionality permitting user input (though notably other commands do exist that are sent between nodes automatically, as can be seen in the command lines of server nodes). As a result of this system, the client is forbidden from entering any command using the characters “@”, “#” or “\*” in submitted queries.

# Load Balancing

When a client submits a request that cannot be processed by a control node, the control node will attempt to route the request to an existing node. If an existing node does not exist or has reached its maximum client limit, the service will attempt to spawn a new node to handle this request. Once a client has been connected to a node, the node will register the commands it is able to process to the client, and the client will henceforth automatically route any of these commands to the node directly. A control node will keep active communication with all nodes it has spawned, allowing them to send updates on how many client connections they have to the control node – if a service that once had reached its maximum number of clients loses a connection, then the control node will resume sending clients to that node until it once again reports meeting a maximum number of clients.

Similarly, the control nodes themselves enforce a client capacity, with the intention of routing connected clients towards another node if the capacity of clients is reached by the control node. This system has one caveat – the control node client capacity is designed as a loose measure, and a control node that is unable to pass along clients to another node (either through other control nodes not existing or all other control nodes being full) will accept client connections past its maximum client capacity.

Finally, one last uniquity to the system designed to allow the best user experience possible comes in the acquisition of files. While a client will normally only have connection to one control node, and only have access to the nodes spawned by said control node, the file distribution system has permission to deviate from this system if a requested file is not available on the currently connected control node. In this situation, the control node handling the clients requests will submit a redirect request to whichever control node has the file in question, which will spawn up a new file handler node and provide it the client data without the foreign control node registering the user data.

This means that the client node will connect to a file distribution node outside the scope of its handling control node, and the control node that does handle this file distribution node will not create a connection with the client. This foreign control node will therefore not calculate this client towards its capacity of clients connected, but the foreign file distribution node will, and its handling control node will spawn up new file distribution nodes as appropriate.