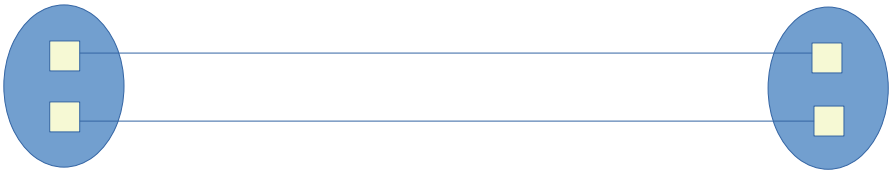
Documentation

server.py

There are 2 ports open for the client, one checks if the client is up and the other one exchanges messages and commands with the client.



SERVER CLIENT

find_local_ip → as we know the only way to have an ip that doesn't constantly change is to purchase one. The authors of the project did not have a static public ip , therefore local ip was used. In order to find the local ip we connect to a web server and via that we can obtain with what ip we connected.

handle_client → This function handles the messages exchanged between the client and the server. In case the client and the server are connected, we can make a choice about what message to send to the client. The input is only a number. The client sends back the message that we check, if the message is :

- "!DISCONNECT" that means the connection will be disconnected.
- "GIVE_NAME" means that we need to provide the name of the wanted file.
- "MAKE_CHOICE" asks us which file we want from the returned list of files from the client.
- "FILE" means the bits of a pickle of is going to be sent.
- "SCREENSHOT" tells the client to take a screenshot.

- “MOVE_MOUSE” asks us for two coordinates and moves the mouse wherever the given coordinates are.

- “SHELL” gives us access to the shell of the computer.

Start → the server starts listening for any connection, all connections are threaded.

handle_client_heartbeat → the client is supposed to tell us that it’s up every minute, this function handles the messages from the heartbeat.
If the message is:

- “!DISCONNECT” that means the connection will be disconnected.

second_start → starts the second server that listens to the heartbeat.

client.py

send_heartbeat → tells the server it is up.
schedule_hearbeat → schedules the heartbeat to be sent every minute.
find_all → finds all files and folders with the given name.
Send → sends the message along with it’s length.
second_send → sends the message along it’s length to the second port

antivirus.py

finds all the active ports with a tcp connection, creates thread for each port and monitors it’s traffic for 30 seconds, and blocks the port if it sends more than 5 requests.

make_threads → creates a thread for every port and executes it.
handle_packet → prints the packet data
find → monitors the connection to each port, in case there are more than 5 packets received within 30 seconds, the port is closed.

unit_tests.py

we tried to write tests, not very successful.