CS 550 - Spring 2022
Isaias Rivera
A20442116

P2P File Sharing - Super-peers - Manual

Note

This manual only covers how to use the program. Terms that will be not explained in this manual are `ip address`, `ports`, `port forwarding`, `directories`, and `file hashes`. If this program is only needed to run locally, the *Simple* examples should work just fine. However, this manual is also mainly for Windows, meaning, the exact syntax of some commands may depend on your system.

Procedure

The typical procedure for using this program is as follows

1. Create valid folders for each client
2. Move files to folders for each client that want to be shared
3. Create the static connection config (or use the one provided)
4. Start at least two clients, using the folders previously created, and where at least one of the clients is a super-peer
5. On one client, search for file
6. Request file, given the file hash (The long string of characters) from searching
7. Wait for file download or repeat steps 5-6
8. Close all clients

Program arguments

A <int> just means it should be a number.

Client

Usage:

Client [-ah] [–version] [-c <filePath>] [-f <directory>] [-i <int>]

Where:

-i <int>, –identity <int> Unique ID identifying this client

-c <filePath>, –configFile <filePath> The config file to use

-f <directory>, –downloadFolder <directory> The local folder files should be uploaded and downloaded to

-a, –all2all enable all2all mode

Example

Simple - Local Only

```
.\Client.exe -i 0 -c "test_config.json" -f "watchFolder"
```

Using the Interactive Console

The main interface with this program involves using the terminal of your device when using the client.

The following options are allowed when using the client.

- `ping`
  - Ping a client's superpeer's indexing server to check it's response time
- `list`
  - List all the files available on the network, meaning, show all the files we can either download or already have
- `search [query]`
  - Search for the query as a substring the in the name of all the files on the network, meaning, search for a file where `query` is what you want to look for
  - Example: `search am`
    * Search for files with an `am` in it's name
- `request [hash]`
  - Request a specific file, given the hash, meaning, copy the `hash` (The long string of characters) from a single file when you did either a `search` or `list` and paste it where `hash` is.
  - Example: `request E2D0FE1585A63EC6009C8016FF8DDA8B17719A637405A4E23C0FF81339148249`
- `q` or `quit` or `exit`
  - Stop the interactive console, this will close the client

Remember, to add or remove files for sharing. Simply delete or add them to your folder that you initially created for the client.

Overall Example

This example will demonstrate following the procedures for a *simple* local case.

Topology

Because this is a simple example, it will only use two super-peers and two leaf-nodes, layed out as such
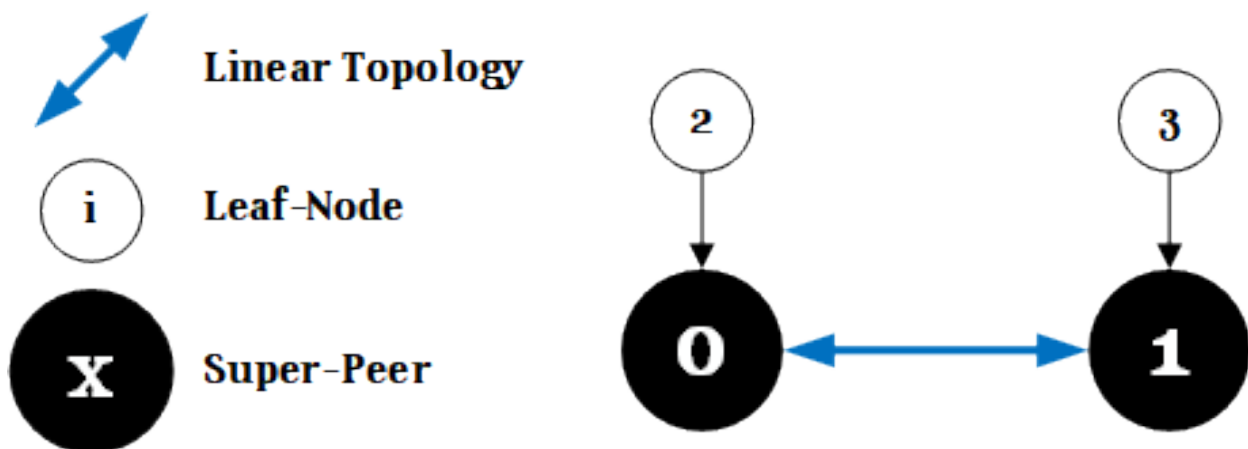
The config file for this is included.

Initial setup

Here we can see where the directories are, relative to the client program.
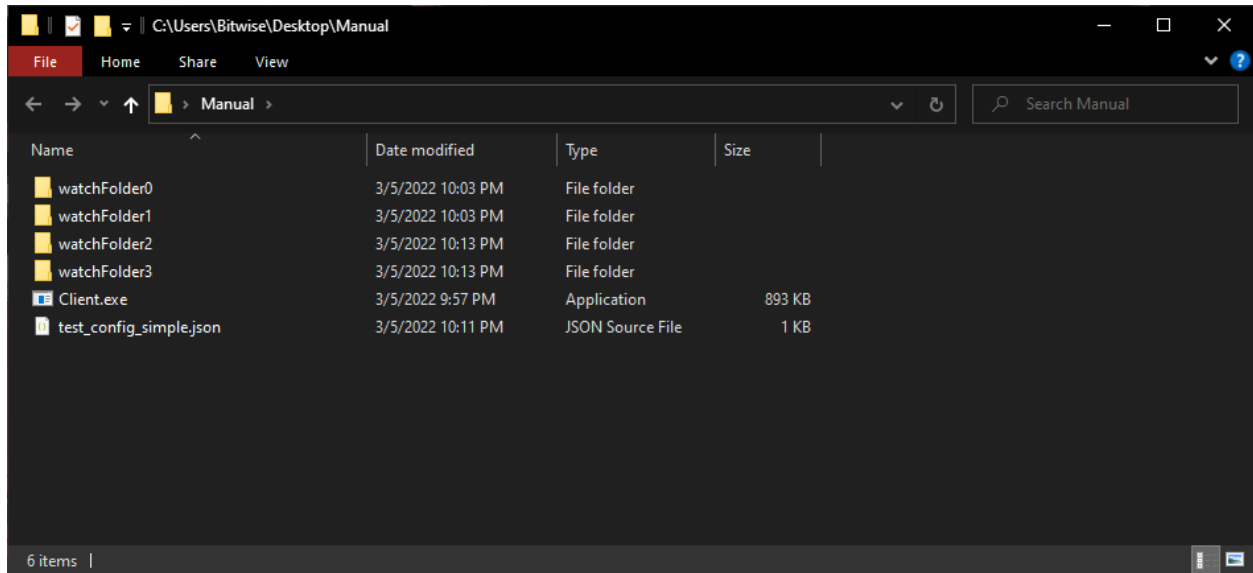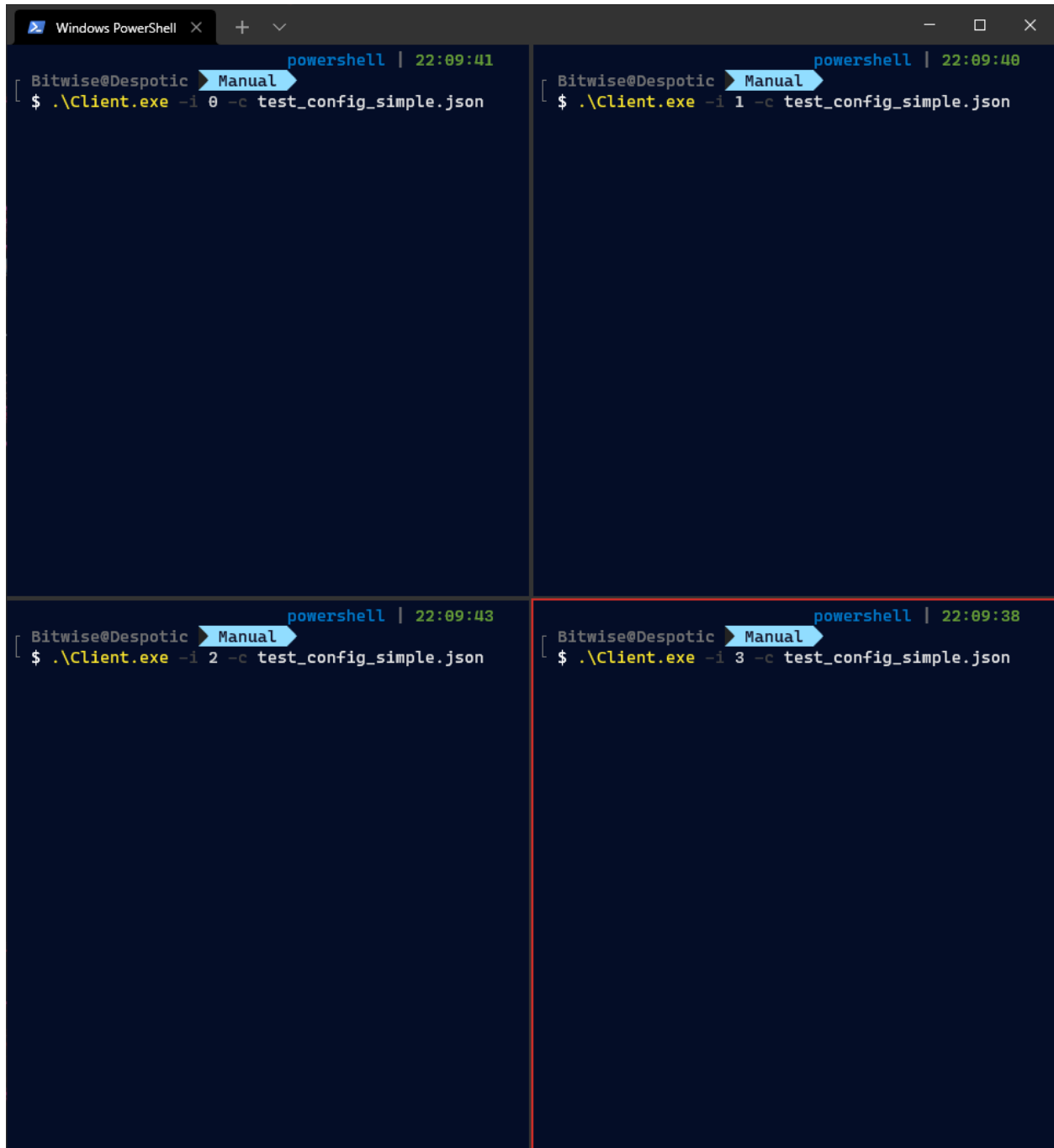
And here we can see the files in each directory
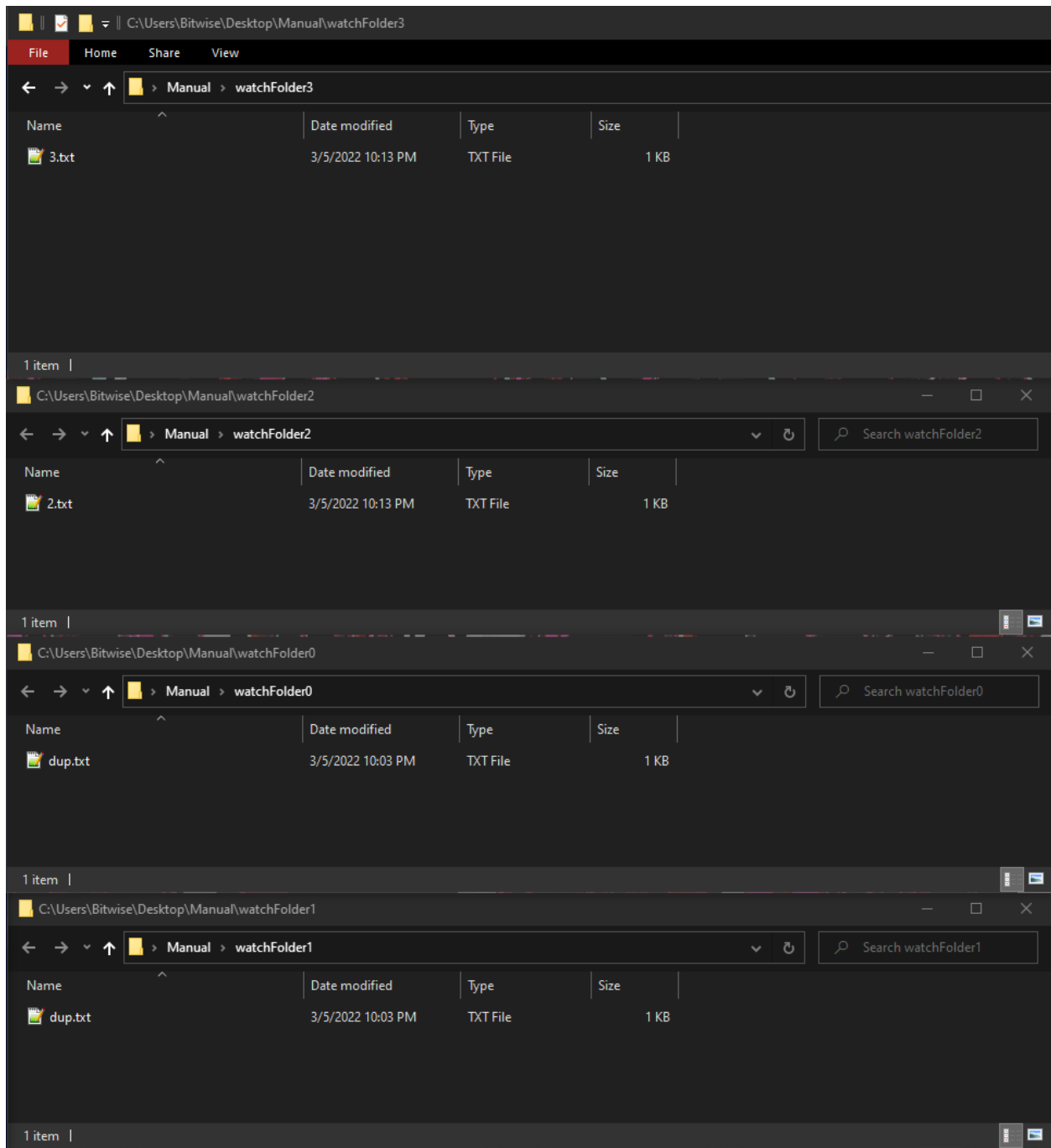
And just as a tip, you can shift right click on Windows in the same directory as the program to more easily open a terminal.

Start Programs

Here we can see the commands are setup to be run.

.\Client.exe -i 0 -c test_config_simple.json start super-peer with id 0
.\Client.exe -i 1 -c test_config_simple.json start super-peer with id 1
.\Client.exe -i 2 -c test_config_simple.json start leaf-node with id 2
.\Client.exe -i 3 -c test_config_simple.json start leaf-node with id 3

NOTE: if no argument is given for the watchfolder it defaults to `watchfolderID` where `ID` is the id of the client.

```
Windows PowerShell  X    +  ∨                                                    —  □  ✕

⌐ Bitwise@Despotic ▶ Manual                  ⌐ Bitwise@Despotic ▶ Manual
└ $ .\Client.exe -i 0 -c test_config_simple.json    └ $ .\Client.exe -i 1 -c test_config_simple.json
[Indexer] INFO    : Running Server              [Indexer] INFO    : Running Server
[Indexer] INFO    : Running Client              [Indexer] INFO    : Running Client
[Indexer] DEBUG   : Connecting to index server at:  [Indexer] DEBUG   : Connecting to index server at:
127.0.0.1:47900                                 127.0.0.1:47901
[Indexer] INFO    : Server pinged! 0.107000ms   [Indexer] INFO    : Server pinged! 0.081000ms
[Exchanger] DEBUG : Listening to port: 48900    [Exchanger] DEBUG : Listening to port: 48901
[Console]         : Console start!              [Console]         : Console start!
 [FileClient-0 > ] DEBUG  : Hashing file: dup.txt  [FileClient-1 > ] DEBUG  : Hashing file: dup.txt
[Folder] DEBUG  : File created: dup.txt         [Folder] DEBUG  : File created: dup.txt
[Registry]          : New                       [Registry]          : New
        ID: 0                                           ID: 1
        name: dup.txt                                   name: dup.txt
        hash: 395A71419DADACF6653548C449A207536831     hash: 395A71419DADACF6653548C449A207536831
AF86ECF265E2067E0377A14A447F                    AF86ECF265E2067E0377A14A447F

[Peer] INFO    : Registered hash: dup.txt       [Peer] INFO    : Registered hash: dup.txt
[Registry]          : New                       [Registry]          : New
        ID: 2                                           ID: 3
        name: 2.txt                                     name: 3.txt
        hash: 8C8C5354325CFD52914A7B36D8A28695F5DE     hash: 76BD6080BE28BDF33FA030A29A6D65849C26
DCBBABEE7C13BE2E02B0CAEF5A6D                    84F2F93DC3A41D3668801DF29FB1


 Client-0 >                                      Client-1 >
─────────────────────────────────────────────────────────────────────────────────
                  powershell | 22:09:43                          powershell | 22:09:38
⌐ Bitwise@Despotic ▶ Manual                  ⌐ Bitwise@Despotic ▶ Manual
└ $ .\Client.exe -i 2 -c test_config_simple.json    └ $ .\Client.exe -i 3 -c test_config_simple.json
[Indexer] INFO    : Running Client              [Indexer] INFO    : Running Client
[Indexer] DEBUG   : Connecting to index server at:  [Indexer] DEBUG   : Connecting to index server at:
127.0.0.1:47900                                 127.0.0.1:47901
[Indexer] INFO    : Server pinged! 0.079000ms   [Indexer] INFO    : Server pinged! 0.080000ms
[Exchanger] DEBUG : Listening to port: 48902    [Exchanger] DEBUG : Listening to port: 48903
[Console]         : Console start!              [Console]         : Console start!
 [File] Client-2 > DEBUG  : Hashing file: 2.txt  [ Client-3File] DEBUG  : > Hashing file: 3.txt
[Folder] DEBUG  : File created: 2.txt           [Folder] DEBUG  : File created: 3.txt
[Peer] INFO    : Registered hash: 2.txt         [Peer] INFO    : Registered hash: 3.txt

 Client-2 >                                      Client-3 >
```

After running them all, you might notice the prompt `Client-x >` becomes a bit garbled, simply pressing `enter` once should fix it. Also, take note that the `x` is the id of the client.

Listing all files

Here we are listing all the available files from client 3

We run the command `list`

Take notice of the `Hash` associated with each file listed, this is what we use to request a file to download

Lets focus on the file `dup.txt` which has the hash of 395A71419DADACF6653548C449A207536831AF86ECF265E2067EC

Requesting a file

Now we need to actually request this file by running the following command from client 3

We run the command `request 395A71419DADACF6653548C449A207536831AF86ECF265E2067E0377A14A447F`

After this runs, we can now see that the file is in the watchfolder for client 3.
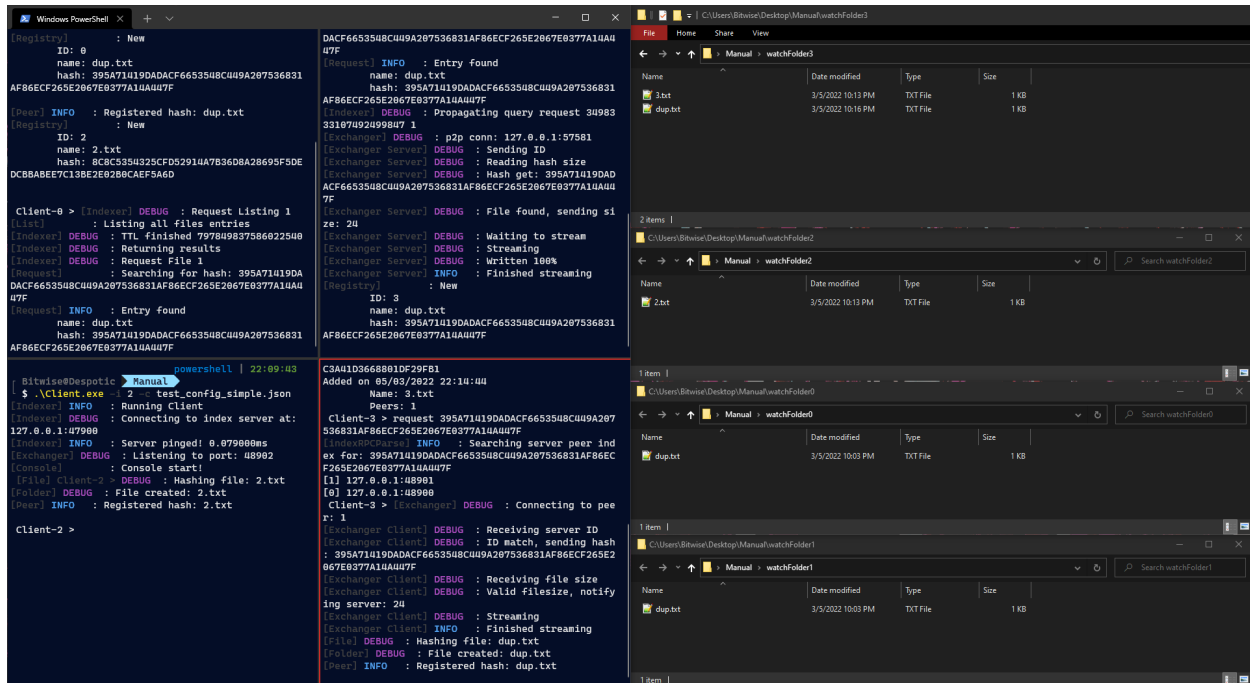
Search for a file

Here we can now see, after searching for `dup` from client 3, that there are now three peers which have the same file.

We run the command `search dup`

Super-peer listing all files

Here we can see super-peer 0 listing all the available files

We run the command `list`

Super-peer requesting a file

Here we can see super-peer 1 request the file `2.txt` using it's appropriate hash

We run the command `request  8C8C5354325CFD52914A7B36D8A28695F5DEDCBBABEE7C13BE2E02B0CAEF5A6D`

After this runs, we can also now see that the file is in the watchfolder for client 1.

Closing

We now can close all clients with `q`.

Here, everything has now closed and we are done.