

P2P File Sharing - Consistency - 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's origin folder 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 `[-ahls] [-version] [-c <filePath>] [-d <directory>] [-i <int>] [-r <int>] [-u <directory>]`

Where:

- `-i <int>`, `-identity <int>` Unique ID identifying this client
- `-c <filePath>`, `-configFile <filePath>` The config file to use
- `-u <directory>`, `-uploadFolder <directory>` The local folder files should be uploaded from
- `-d <directory>`, `-downloadFolder <directory>` The local folder files should be downloaded to
- `-r <int>`, `-ttr <int>` Time To Refresh (TTR) in seconds, if in pulling mode
- `-a`, `-all2all` enable all2all mode
- `-s`, `-pushing` enable pushing of invalidation calls
- `-l`, `-pulling` enable pulling of invalidation calls

Example

Simple - Local Only

```
.\Client.exe -i 0 -s -c "test_config.json" -u "watchFolder" -d "downloadfolder"
```

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
- **refresh**
 - If in pulling mode, this command forces a refresh on all invalid files
- **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

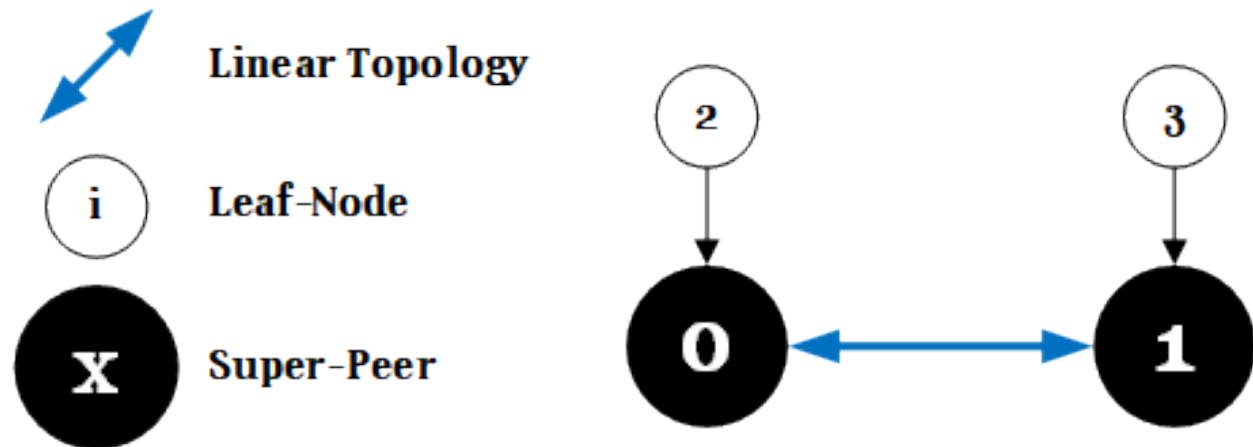
Remember, to add or remove files for sharing. Simply delete or add them to your origin folder that you initially created for the client.

Overall Example

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

Topology

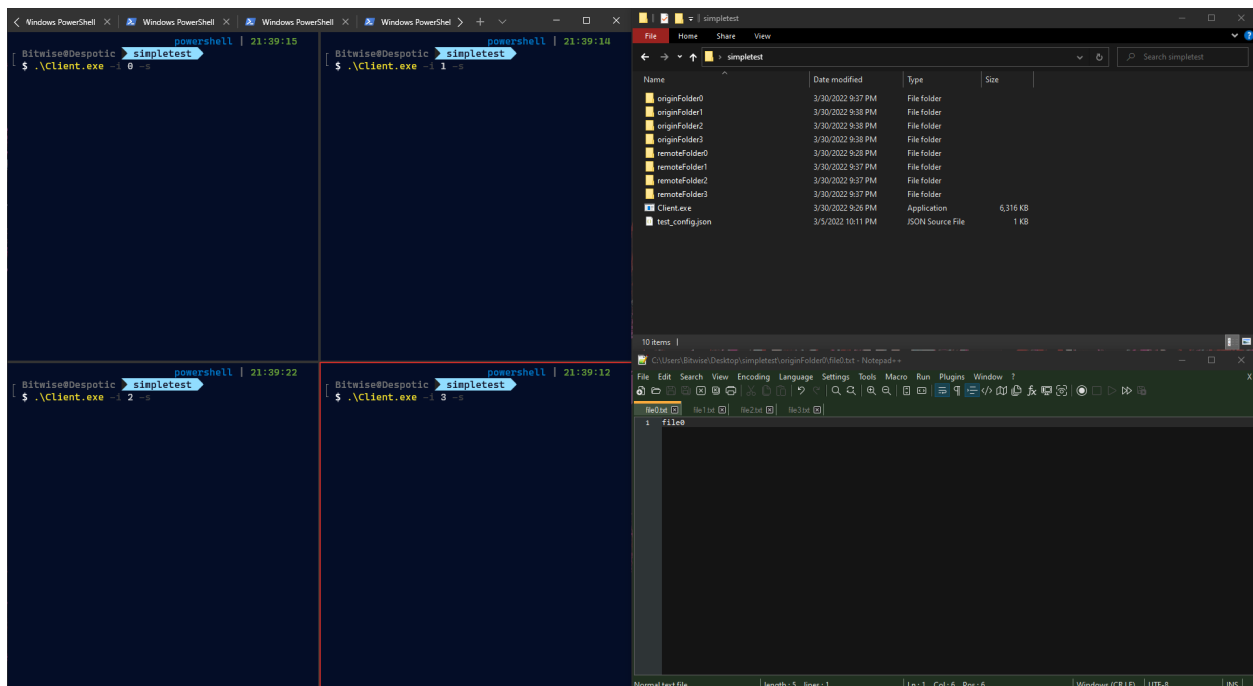
Because this is a simple example, it will only use two super-peers and two leaf-nodes, layed out as such



The config file for this is included.

Initial setup

Here we can see the terminal is split into four for each peer. The folders relative to the app. And the files that are in each origin folder (Only file0 is shown open but all the other files also have unique data).



```

.\Client.exe -i 0 -s start super-peer with id 0 in push mode
.\Client.exe -i 1 -s start super-peer with id 1 in push mode
.\Client.exe -i 2 -s start leaf-node with id 2 in push mode
.\Client.exe -i 3 -s start leaf-node with id 3 in push mode

```

NOTE: if no argument is given for the watchfolder it defaults to XID where X is either originFolder and remoteFolder and where ID is the id of the client.

Running Program

```

ID: 0
name: file0.txt
hash: 56F3FD843F7AE959A809E0AE7C067A0E862
A6FAA7A22AD147EE90EE5992BD7

[Registry] : New
ID: 0
name: file0.txt
hash: 56F3FD843F7AE959A809E0AE7C067A0E862
A6FAA7A22AD147EE90EE5992BD7

[Peer] INFO : Registered hash: file0.txt
[Registry] INFO : New Origin
ID: 2
name: file2.txt
hash: 33778780FEAAA7ADF79A374D2702A3FD813E
9E5EA0D08AA95A882AD39844A92F

[Registry] : New
ID: 2
name: file2.txt
hash: 33778780FEAAA7ADF79A374D2702A3FD813E
9E5EA0D08AA95A882AD39844A92F

Client-0 >

ID: 1
name: file1.txt
hash: C147EFCFC2D7EA666A9E4F5187B115C98903
F8FC896A56DF9A6EF5D8F3FC9F31

[Registry] : New
ID: 1
name: file1.txt
hash: C147EFCFC2D7EA666A9E4F5187B115C98903
F8FC896A56DF9A6EF5D8F3FC9F31

[Peer] INFO : Registered hash: file1.txt
[Registry] INFO : New Origin
ID: 3
name: file3.txt
hash: 6F3FEF6DC51C7996A749928780C35F328ED
989A5E97646CF8BA83383C958B02

[Registry] : New
ID: 3
name: file3.txt
hash: 6F3FEF6DC51C7996A749928780C35F328ED
989A5E97646CF8BA83383C958B02

Client-1 >

ID: 2
name: file2.txt
hash: 33778780FEAAA7ADF79A374D2702A3FD813E
9E5EA0D08AA95A882AD39844A92F

[Registry] : New
ID: 2
name: file2.txt
hash: 33778780FEAAA7ADF79A374D2702A3FD813E
9E5EA0D08AA95A882AD39844A92F

Client-2 >

ID: 3
name: file3.txt
hash: 6F3FEF6DC51C7996A749928780C35F328ED
989A5E97646CF8BA83383C958B02

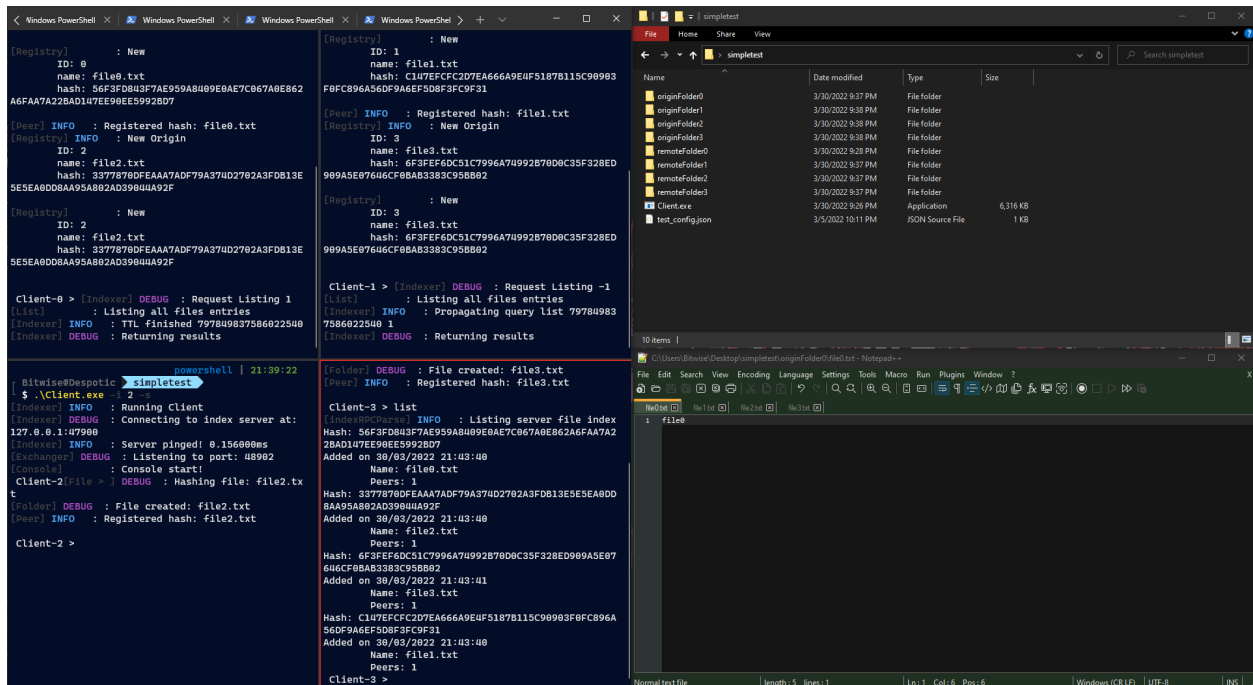
[Registry] : New
ID: 3
name: file3.txt
hash: 6F3FEF6DC51C7996A749928780C35F328ED
989A5E97646CF8BA83383C958B02

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.

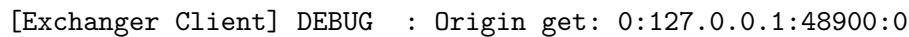
We run the command `list`



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 `file0.txt` which has the hash of `56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147`

After this runs, we can now see that the file is in the remote folder for client 3.



Search for a file

Here we can now see, after searching for the term file from client 2, that there are now two peers which have the same file.

We run the command `search file`

```
Windows PowerShell x Windows PowerShell x Windows PowerShell x Windows PowerShell > + v - □ x

[Indexer] DEBUG : Request File 1
[Request] : Searching for hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Request] INFO : Entry found
            name: file0.txt
            hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Exchanger] DEBUG : p2p conn: 127.0.0.1:58177
[Exchanger Server] DEBUG : P2PFile
[Exchanger Server] DEBUG : Sending ID
[Exchanger Server] DEBUG : Hash size get: 64
[Exchanger Server] DEBUG : Hash get: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Exchanger Server] DEBUG : File found, sending size: 5
[Exchanger Server] DEBUG : Waiting to stream
[Exchanger Server] DEBUG : Streaming
[Exchanger Server] DEBUG : Written 100%
[Exchanger Server] INFO : Finished streaming
[Indexer] DEBUG : Request Search -1
[Search] : Searching for query: file
[Indexer] INFO : Propagating query search 1232841491290556117 1
[Indexer] DEBUG : Returning results

Client-1 > [Indexer] DEBUG : Request Listing -1
[List] : Listing all files entries
[Indexer] INFO : Propagating query list 797849837586022540 1
[Indexer] DEBUG : Returning results
[Indexer] DEBUG : Request File -1
[Request] : Searching for hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Request] WARNING: No entries found with hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Indexer] INFO : Propagating query request 3498333107492499847 1
[Registry] : New
            ID: 3
            name: file0.txt
            hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Entry] DEBUG : TTR Updated: file0.txt:1648694786
[Indexer] DEBUG : Request Search 1
[Search] : Searching for query: file
[Indexer] INFO : TTL finished 1232841491290556117
[Indexer] DEBUG : Returning results

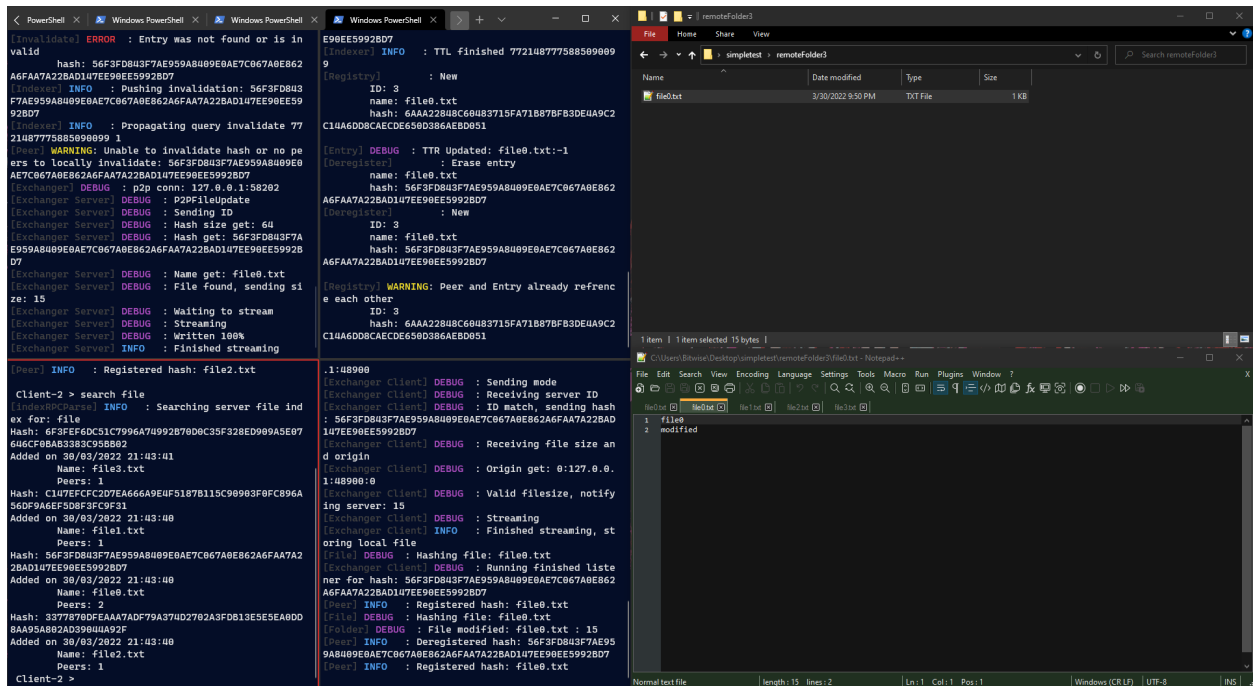
[Peer] INFO : Registered hash: file2.txt

Client-2 > search file
[indexRPCParse] INFO : Searching server file index for: file
Hash: 6F3FEF6DC51C7996A74992B70D0C35F328ED909A5E07646CF0B8B3383C95BB02
Added on 30/03/2022 21:43:41
            Name: file3.txt
            Peers: 1
Hash: C147EFCFC2D7EA666A9E4F5187B115C90903F0FC896A56DF9A6EF5D8F3FC9F31
Added on 30/03/2022 21:43:40
            Name: file1.txt
            Peers: 1
Hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
Added on 30/03/2022 21:43:40
            Name: file0.txt
            Peers: 2
Hash: 3377870DFEAAA7ADF79A374D2702A3FDB13E5E5EA0DD8AA95A802AD39044A92F
Added on 30/03/2022 21:43:40
            Name: file2.txt
            Peers: 1
Client-2 >

A22BAD147EE90EE5992BD7
[0] 127.0.0.1:48900
Client-3 > [Exchanger] DEBUG : Connecting to peer: 0:127.0.0.1:48900
[Exchanger Client] DEBUG : Sending mode
[Exchanger Client] DEBUG : Receiving server ID
[Exchanger Client] DEBUG : ID match, sending hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Exchanger Client] DEBUG : Receiving file size and origin
[Exchanger Client] DEBUG : Origin get: 0:127.0.0.1:48900:0
[Exchanger Client] DEBUG : Valid filesize, notifying server: 5
[Exchanger Client] DEBUG : Streaming
[Exchanger Client] INFO : Finished streaming, starting local file
[File] DEBUG : Hashing file: file0.txt
[Exchanger Client] DEBUG : Running finished listener for hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt
[File] DEBUG : Hashing file: file0.txt
[Folder] DEBUG : File created: file0.txt
```

Modify file

Here we can see super-peer 0 propagated an invalidation after modifying it's file which caused peer 3 to re-request the file from peer 0.



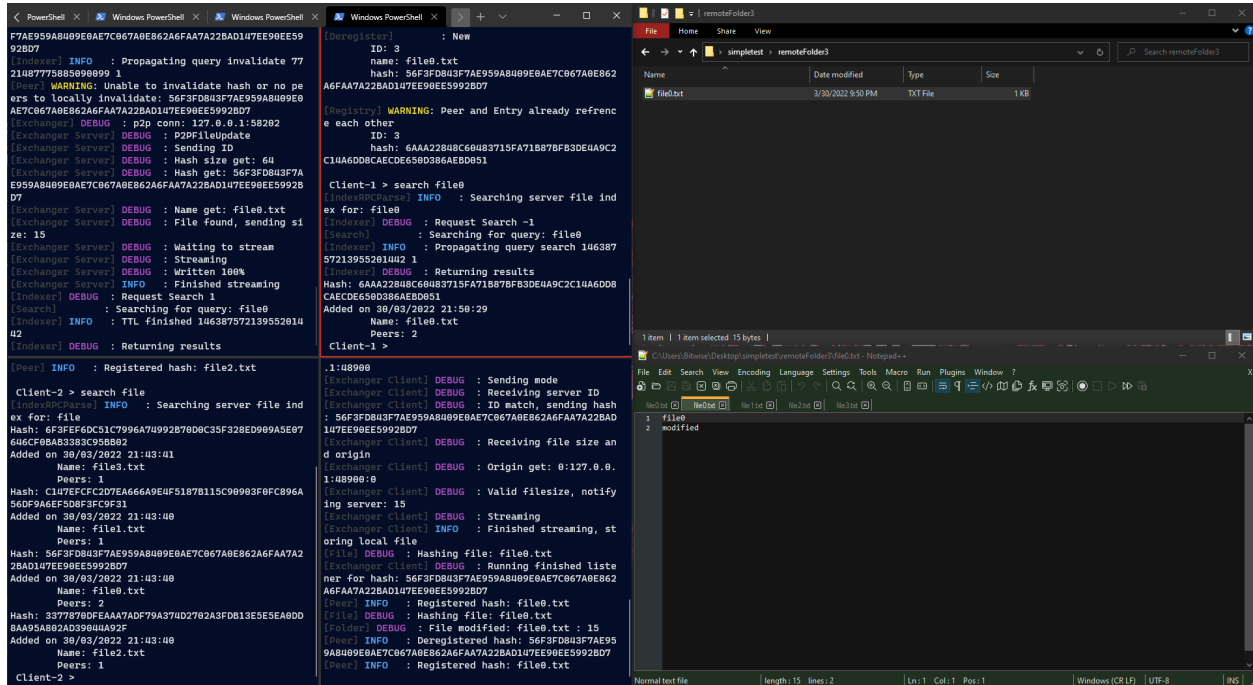
```
[Invalidation] ERROR : Entry was not found or is in vaId
hash: 56F3FD843F7AE959A8489E0AE7C067A0E862
AGFAA7A22BAD147EE90EE5992BD7
[Invalidation] INFO : Pushing invalidation: 56F3FD843
F7AE959A8489E0AE7C067A0E862AGFAA7A22BAD147EE90EE59
92BD7
[Invalidation] INFO : Propagating query invalidate 77
2148777588509099
[Invalidation] WARNING: Unable to invalidate hash or no pe
ers to locally invalidate: 56F3FD843F7AE959A8489E0
AE7C067A0E862AGFAA7A22BAD147EE90EE5992BD7
[Exchanger Server] DEBUG : p2p conn: 127.0.0.1:58282
[Exchanger Server] DEBUG : P2PFileUpdate
[Exchanger Server] DEBUG : Sending ID
[Exchanger Server] DEBUG : Hash size get: 64
[Exchanger Server] DEBUG : Hash get: 56F3FD843F7A
E959A8489E0AE7C067A0E862AGFAA7A22BAD147EE90EE5992B
D7
[Exchanger Server] DEBUG : Name get: file0.txt
[Exchanger Server] DEBUG : File found, sending si
ze: 15
[Exchanger Server] DEBUG : Waiting to stream
[Exchanger Server] DEBUG : Streaming
[Exchanger Server] DEBUG : Written 100%
[Exchanger Server] INFO : Finished streaming
[Peer] INFO : Registered hash: file2.txt
Client-2 > search file
[Invalidation] INFO : Searching server file ind
ex for: file
Hash: 6F3FE6DC51C7996A70992B70D0C35F328ED090A5E07
60CF0BA83383C958802
Added on 30/03/2022 21:43:41
Name: file3.txt
Peers: 1
Hash: C147EFCFC2D7EA666A9E4F5187B115C90903F0FCB96A
56DF9A6EF508F3FC9F31
Added on 30/03/2022 21:43:40
Name: file1.txt
Peers: 1
Hash: 56F3FD843F7AE959A8489E0AE7C067A0E862AGFAA7A2
2BAD147EE90EE5992BD7
Added on 30/03/2022 21:43:40
Name: file0.txt
Peers: 2
Hash: 3277870DEFAA7ADF79A374D2702A3F0B13E5E5EA0D0
8A95A802A03900A925
Added on 30/03/2022 21:43:40
Name: file2.txt
Peers: 1
Client-2 >
E90EE5992BD7
[Invalidation] INFO : TTL finished 77214877588509099
9
[Registry] : New
ID: 3
name: file0.txt
hash: 4AAA22848C60483715FA71887BF83DE4A9C2
C14A6D08CAECDE650D386AEB051
[Entry] DEBUG : TTR Updated: file0.txt:-1
[DeRegister] : Erase entry
name: file0.txt
hash: 56F3FD843F7AE959A8489E0AE7C067A0E862
AGFAA7A22BAD147EE90EE5992BD7
[DeRegister] : New
ID: 3
name: file0.txt
hash: 56F3FD843F7AE959A8489E0AE7C067A0E862
AGFAA7A22BAD147EE90EE5992BD7
[Registry] WARNING: Peer and Entry already referenc
e each other
ID: 3
hash: 6AAA22848C60483715FA71887BF83DE4A9C2
C14A6D08CAECDE650D386AEB051
.1:48900
[Exchanger Client] DEBUG : Sending mode
[Exchanger Client] DEBUG : Receiving server ID
[Exchanger Client] DEBUG : ID match, sending hash
: 56F3FD843F7AE959A8489E0AE7C067A0E862AGFAA7A22BAD
147EE90EE5992BD7
[Exchanger Client] DEBUG : Receiving file size an
d origin
[Exchanger Client] DEBUG : Origin get: 0:127.0.0.
1:48900:0
[Exchanger Client] DEBUG : Valid filesize, notify
ing server: 15
[Exchanger Client] DEBUG : Streaming
[Exchanger Client] INFO : Finished streaming, st
oring local file
[File] DEBUG : Hashing file: file0.txt
[Exchanger Client] DEBUG : Running finished liste
ner for hash: 56F3FD843F7AE959A8489E0AE7C067A0E862
AGFAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt
[File] DEBUG : Hashing file: file0.txt
[Folder] DEBUG : File modified: file0.txt : 15
[Peer] INFO : Deregistered hash: 56F3FD843F7AE95
9A8489E0AE7C067A0E862AGFAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt
```

note that the file open is not the original file, it is the file located in peer 3's remote folder

Search for modified file

Here we can now see, after searching for the term `file0` from client 1, that there are still two peers which have the same file.

We run the command `search file0`



```
[Peer] INFO : Registered hash: file2.txt
Client-2 > search file
[Indexer] INFO : Searching server file index for: file
Hash: 6F3FE6DC51C7996A74992B70DC35F328ED09A5E07
60CF08A38383958862
Added on 30/03/2022 21:43:41
Name: file3.txt
Peers: 1
Hash: C147EFCFC2D7EA666A9E4F5187B115C99903F0FC896A
56DF9A6EF508F3C9F31
Added on 30/03/2022 21:43:40
Name: file1.txt
Peers: 1
Hash: 56F3FD843F7AE959A809E0AE7C067A0E862A6FAA7A2
2BAD147EE90EE5992BD7
Added on 30/03/2022 21:43:40
Name: file0.txt
Peers: 2
Hash: 3377870DFAAA7ADF79A374D2702A3FD813E5E5EA0DD
8AA95A802AD3904A92F
Added on 30/03/2022 21:43:40
Name: file2.txt
Peers: 1
Client-2 >
```

```
[Deregister] : New
ID: 3
name: file0.txt
hash: 56F3FD843F7AE959A809E0AE7C067A0E862
A6FAA7A22BAD147EE90EE5992BD7
[Registry] WARNING: Peer and Entry already reference each other
ID: 3
hash: 6AAA2284BC60483715FA71B87BF83DE4A9C2
C14A6D08CAECDE650D386AED051
Client-1 > search file0
[Indexer] INFO : Searching server file index for: file0
[Indexer] DEBUG : Request Search-1
[Search] : Searching for query: file0
[Indexer] INFO : Propagating query search 146387
57213955201042 1
[Server] DEBUG : Returning results
Hash: 6AAA2284BC60483715FA71B87BF83DE4A9C2C14A6D08
CAECDE650D386AED051
Added on 30/03/2022 21:58:29
Name: file0.txt
Peers: 2
Client-1 >
```

```
[Exchanger Client] DEBUG : Sending mode
[Exchanger Client] DEBUG : Receiving server ID
[Exchanger Client] DEBUG : ID match, sending hash
: 56F3FD843F7AE959A809E0AE7C067A0E862A6FAA7A22BAD
147EE90EE5992BD7
[Exchanger Client] DEBUG : Receiving file size and origin
[Exchanger Client] DEBUG : Origin get: 0:127.0.0.
1:00000:0
[Exchanger Client] DEBUG : Valid filesize, notifying server: 15
[Exchanger Client] DEBUG : Streaming
[Exchanger Client] INFO : Finished streaming, storing local file
[File] DEBUG : Hashing file: file0.txt
[Exchanger Client] DEBUG : Running finished listner for hash: 56F3FD843F7AE959A809E0AE7C067A0E862
A6FAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt
[File] DEBUG : Hashing file: file0.txt
[Folder] DEBUG : File modified: file0.txt : 15
[Peer] INFO : Deregistered hash: 56F3FD843F7AE95
9A809E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt
```

Closing

We now can close all clients with q and we are done.

```
< PowerShell x Windows PowerShell x Windows PowerShell x Windows PowerShell x + - □ x
D7
[Exchanger Server] DEBUG : Name get: file0.txt
[Exchanger Server] DEBUG : File found, sending size: 15
[Exchanger Server] DEBUG : Waiting to stream
[Exchanger Server] DEBUG : Streaming
[Exchanger Server] DEBUG : Written 100%
[Exchanger Server] INFO : Finished streaming
[Indexer] DEBUG : Request Search 1
[Search] : Searching for query: file0
[Indexer] INFO : TTL finished 14638757213955201442
[Indexer] DEBUG : Returning results

Client-0 > q
[Console] INFO : Exiting
[Folder] DEBUG : Stopped watching path for changes
[Folder] DEBUG : Stopped watching path for changes
[Exchanger] DEBUG : stopped receiving peer requests
[Indexer] INFO : Stopping Client
powershell | 21:54:03
[ Bitwise@Despotic simpletest
$

56DF9A6EF5D8F3FC9F31
Added on 30/03/2022 21:43:40
Name: file1.txt
Peers: 1
Hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
Added on 30/03/2022 21:43:40
Name: file0.txt
Peers: 2
Hash: 3377870DFEAAA7ADF79A374D2702A3FDB13E5E5EA0DD8AA95A802AD39044A92F
Added on 30/03/2022 21:43:40
Name: file2.txt
Peers: 1

Client-2 > q
[Console] INFO : Exiting
[Folder] DEBUG : Stopped watching path for changes
[Folder] DEBUG : Stopped watching path for changes
[Exchanger] DEBUG : stopped receiving peer requests
[Indexer] INFO : Stopping Client
powershell | 21:54:01
[ Bitwise@Despotic simpletest
$

Client-1 > search file0
[indexRPCParse] INFO : Searching server file index for: file0
[Indexer] DEBUG : Request Search -1
[Search] : Searching for query: file0
[Indexer] INFO : Propagating query search 14638757213955201442 1
[Indexer] DEBUG : Returning results
Hash: 6AAA22848C60483715FA71B87BFB3DE4A9C2C14A6DD8CAECDE650D386AEBD051
Added on 30/03/2022 21:50:29
Name: file0.txt
Peers: 2

Client-1 > q
[Console] INFO : Exiting
[Folder] DEBUG : Stopped watching path for changes
[Folder] DEBUG : Stopped watching path for changes
[Exchanger] DEBUG : stopped receiving peer requests
[Indexer] INFO : Stopping Client
powershell | 21:54:03
[ Bitwise@Despotic simpletest
$

[Exchanger Client] DEBUG : Streaming
[Exchanger Client] INFO : Finished streaming, storing local file
[File] DEBUG : Hashing file: file0.txt
[Exchanger Client] DEBUG : Running finished listener for hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt
[File] DEBUG : Hashing file: file0.txt
[Folder] DEBUG : File modified: file0.txt : 15
[Peer] INFO : Deregistered hash: 56F3FD843F7AE959A8409E0AE7C067A0E862A6FAA7A22BAD147EE90EE5992BD7
[Peer] INFO : Registered hash: file0.txt

Client-3 > q
[Console] INFO : Exiting
[Folder] DEBUG : Stopped watching path for changes
[Folder] DEBUG : Stopped watching path for changes
[Exchanger] DEBUG : stopped receiving peer requests
[Indexer] INFO : Stopping Client
powershell | 21:54:04
[ Bitwise@Despotic simpletest
$
```