StoreIt Protocol v0.2

Adrien Morel adrien.morel@me.com

March 23, 2016

1 Introduction

Hello, this is the documentation for the StoreIt protocol. It is used when two clients talk and when the clients and the server talk.
Everything which is not some binary data (chunks) should be encoded in ASCII. Everything in the protocol is ether a *command* or a *numeric response*. A command will always have the following structure:

NAME SIZE *arg1 [arg2 arg3...]* [payload]

- NAME is the command name. "JOIN" for example.
- SIZE is the size in bytes following the ' ' (space) after SIZE. For example *TAKE 9 money 900*.
- *arg* are command ASCII arguments.
- payload is a list of bytes for some payload. Example :

```
STORE 10 hi.txt hello world
```

  Here, "hello world" is the payload and "hi.txt" is an argument.

When I write [client] -> [server] it means "this command is sent by a client to our backend servers". [client] -> [client] means "it is sent from client to client" (p2p).

2 The JSON data structures

2.1 File Object

```
{
  "path": "storeit",
  "metadata": "unimplemented for now",
  "unique_hash": "sha256 hash of all the data in the file",
  "kind": 0,

  // for now, this list is not necessary.
  "chunks_hashes": [
    "some sha256"
  ],
  "files": {
    "file_name": anotherFileObject,
    "another_file_name": anotherFileObject
  }
}
```

3 ASCII Commands

3.1 Session

JOIN [client] -> [server]
This is the first request to make whenever a client wants to get online.

JOIN SIZE username port json_file

Json_file is the json tree describe in the second section.

LEAVE [client] -> [server]
A client should send this to our server to leave the network.

LEAVE 0

Note that '0' is the SIZE parameter. It is always zero since there is no argument.

3.2 Raw Data Manipulation (chunks)

CHDELETE [server] -> [client]
The server asks the client to delete a chunk it is storing.

CHDELETE SIZE chunk_hash

CHSEND [server] -> [client]
Tells a client to wait for a specific client to send it a chunk (only in the case of chunks that don't belongs to a user), or to send a chunk to another client. See the diagram at the end of the document to better understand.

CHSEND SIZE send chunk_hash ip_address:port

- send:
  - 0: should wait for incoming chunk.
  - 1: should send the chunk to the specified client.
- chunk_hash: the chunk's hash
- ip_address:port: the ip address and port of the client that should be contacted.

CHSTORE [client] -> [client]
Send a chunk to a client.

CHSTORE SIZE payload

Payload is the chunk bytes.

3.3 File Management

FDELETE [client] -> [server] or [server] -> [client]
Delete a file/directory.

FDELETE SIZE file_path

FADD [client] -> [server] or [server] -> [client]

Add a file to the user tree. If it is sent from the server, the user should create the file and wait further instructions to fetch the file data. Otherwise if it is sent from the client, the server will attempt to add the file to the user virtual tree. It will then send further instructions to dispatch the data on the network. See the diagram for better understanding.

FADD SIZE json_file

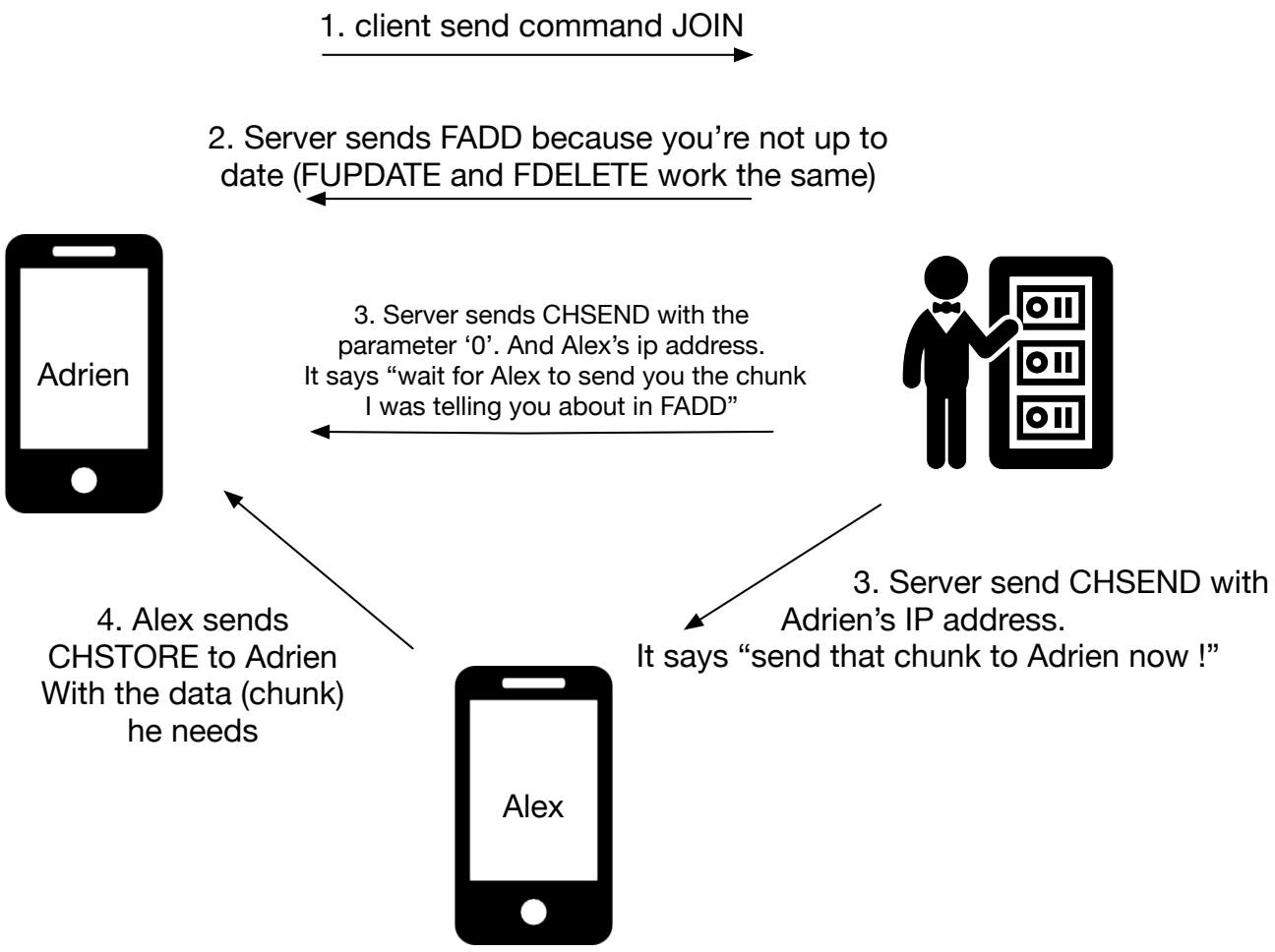
FUPDATE [client] -> [server] or [server] -> [client]
Tell the server to update a file, or receive the order to update a file. Like FADD, it will be followed by further instructions to get the raw data.

FUPDATE SIZE json_file

4 Numeric responses

textbfThis is not yet implemented on the server so don’t implement it for now. Just send ’0 0’ if everything went well (it’s the command ’0’ with size 0. Send ’1’ SIZE ’error message’ if something wrong occured and you could not complete the command.

5 Examples Diagrams


1. client send command JOIN
2. Server sends FADD because you're not up to date (FUPDATE and FDELETE work the same)
Adrien
3. Server sends CHSEND with the parameter '0'. And Alex's ip address. It says "wait for Alex to send you the chunk I was telling you about in FADD"
3. Server send CHSEND with Adrien's IP address. It says "send that chunk to Adrien now !"
4. Alex sends CHSTORE to Adrien With the data (chunk) he needs
Alex