Team

Name	Student ID	Lab group
Eilidh McAlonan	2560102M	LB03
Taylor Douglas	2664501D	LB03

Protocol

These are the valid messages used in our protocol.

- Packet size: 1KiB[1]

Process

- An initial request will be made, specifying which type of request
- An acknowledgement will always be returned from the server, regardless of if the request was invalid
- If the request was valid, the server/client will transmit the data in packets

Formats

Initial request

`type={0,1,2},name={<str>},n={<int>}`

Definitions

- `type`: 0 = get, 1 = put, 2 = list
- `name`: The file name. Will be the empty string if type = 2
- `n`: The number of packets required to upload the file. 0 unless type = 1

Request acknowledgement:

`status={0,1},msg={<str>},n={<int>}`

[1] Kibibytes (KiB) are 1024 bits

Definitions

- `status`: 0 = success, 1 = error
- `msg`: The error message. Will be the empty string if type = 0
- `n`: The number of packets that the file will be sent in. 0 unless type = 0

File packets

- The file encoded in binary format
- Split into packets. Use the `n` header to determine how many packets to listen for

List response

First packet: `n={<int>}`

Next `n` packets: `name1,name2,name3,...`

Design decisions

Inspiration

- We used previous knowledge of the TCP protocol as a basis for our protocol
- Our protocol does a simpler version of the intial handshake of TCP. This inspired the formatting of our handshake messages
- It's also stream-based rather than using datagrams

Cooperation

- We split up `lib.py` and `request.py` to separate concerns
  - `lib.py` handles shared functions like the process of sending and receiving messages
  - `request.py` deals with formatting and parsing messages
  - This meant that `client.py` and `server.py` just had to dispatch to the correct send/receive function without having to understand any of that complexity
  - This modularity helped us to work on different things in parallel
- We used git and GitHub for source control

Issues

- `n` is communicated in 3 different places, depending on the request type
- We did this because it was the simplest idea that we had for communicating how many packets would be used
- It turned out to be inefficient and complicated
- If given the time to redesign the protocol, we would communicate `n` only once - likely in the acknowledges