Midterm#4

Dec. 20th

Write a “parallel file transfer” application where a server accepts connections from clients willing store files. The file transfer will be performed in parallel over multiple TCP connections.

The server, upon startup, creates a socket on a well known port and starts listening for requests from clients.

A client, willing to store/retrieve a file (put/get) forks multiple processes, the number of which is defined in an environment variable.

The forked processes then connect to the server and transfer chunks (a portion of the file for each process) to the remote site.

Example

```
$ export NPIECES=4
$ client put mylargefile
```

This will split and transfer the file in 4 pieces, storing each chunk in a different file on the server.

```
$ client get mylargefile
```

This will receive and reassemble the 4 pieces, from the chunks stored on the server.

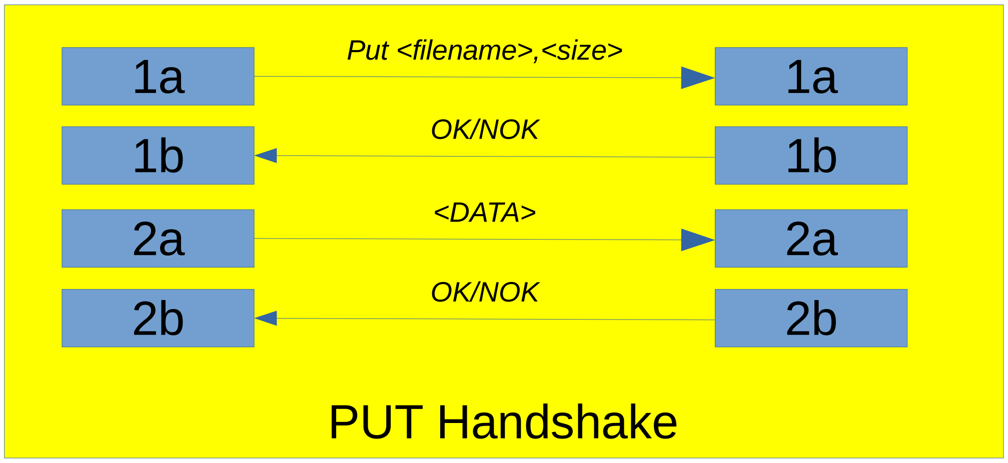
Hints:

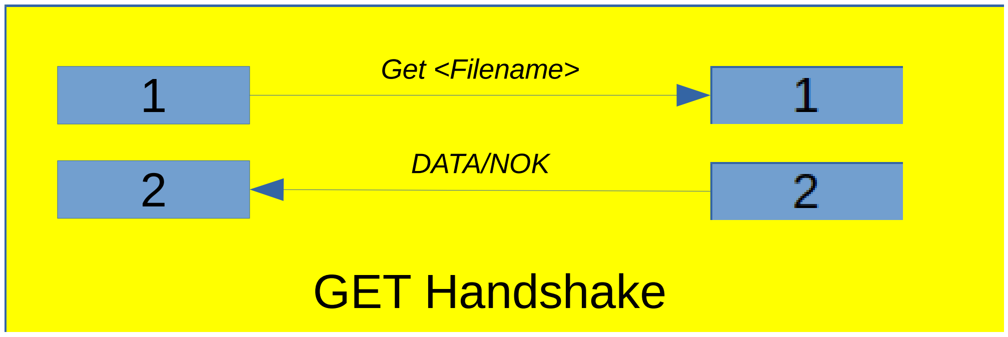
Use the `getenv()` function to retrieve environment variables.

The client could either read the whole file into memory, and then split it for the transfer (dangerous for large files), or combine `open()/stat()/lseek()/read()`. The same goes for transfers in the opposite direction

Remote file names could be managed by clients (e.g. a unique name for each chunk of the original file).

The server could fork a new process for each connection. The new process could get the needed information during the initial part of the transfer (i.e the first packet might contain the filename and the chunk’s size, to avoid the server running out of space).


1a
Put <filename>,<size>
1a
1b
OK/NOK
1b
2a
<DATA>
2a
2b
OK/NOK
2b
PUT Handshake


1
Get <Filename>
1
2
DATA/NOK
2
GET Handshake