Project Malcolm

Authors: Ubadah Jafry, Hamza Ijaz

Working

The whole system works in the following manner

- Server are launched using the `server.py` which launches the virtual servers that are set on the port given in the arguments
- The ports for virtual servers are verified if they are correct or not and will be removed from before starting them
- Client is launched using the `client.py` where the parameters for launched servers are passed
- The client tries to connect with server and receives a checksum from all the server
- The checksum is generated from the file which is going to be downloaded by the client and will **only** connect to the server which have the same checksum in majority. This is a initial corruption check done by the client
- Now the client will request the file size from one of the server instead of all of them since the same checksum guaretee same size as well
- After receiving the size of file, the client divides them into equal parts (such that they are integers) and send argument to the server
- The argument contains offset and bytes to transfer, which will be parsed by server and the specified part of the file will be sent
- The client after receiving the file will generate combine them and generate a checksum and compare to the original checksum. In case, this fails the client will restart the process
- Once the file is checked, the connection is terminated by the client

Files

`server.py`

Executable file that launches multiple virtual servers and display there status after certain interval

Arguments

- `-i --interval` Time interval in seconds between server status reporting in seconds
- `-n --number` Total number of virtual server
- `-f --file` Path to the file (either absolute or relative)
- `-p --port` List of ports that must be equal to number of virtual servers

Examples

```
$ ./server.py --help
usage: server.py [-h] -i INTERVAL -n NUMBER -f FILE -p [PORT [PORT ...]]
```

```
Launches multiple virtual servers and display there status after certain
interval

optional arguments:
  -h, --help            show this help message and exit
  -i INTERVAL, --interval INTERVAL
                        Time interval in seconds between server status
reporting in seconds
  -n NUMBER, --number NUMBER
                        Total number of virtual server
  -f FILE, --file FILE  Path to the file (either absolute or relative)
  -p [PORT [PORT ...]], --port [PORT [PORT ...]]
                        List of ports that must be equal to number of
virtual servers
```

```
$ ./server.py -f data.mp4 -n 4 -i 1 -p 10001 10002 10003 10004
```

```
$ python server.py --file data.mp4 --number 4 --interval 1 --port 10001
10002 10003 10004
```

client.py

Executable file that launches a client that can connect to multiple servers that includes error handling and file verification

Arguments

- `-i --interval` Time interval in seconds between server status reporting in seconds
- `-o --output` Path to the output directory (either absolute or relative)
- `-a --address` The ip address of the server
- `-p --port` List of ports that must be equal to number of virtual servers
- `-r --resume` Flag that tells the client whether to resume the existing download in progress

Examples

```
$ ./client.py --help
usage: client.py [-h] -i INTERVAL -o OUTPUT -a ADDRESS -p [PORT [PORT ...]]
[-r RESUME]

Launches a client that can connect to multiple servers that includes error
handling and file verification

optional arguments:
  -h, --help            show this help message and exit
  -i INTERVAL, --interval INTERVAL
```

```
                        Time interval in seconds between server status
reporting in seconds
  -o OUTPUT, --output OUTPUT
                        Path to the output directory (either absolute or
relative)
  -a ADDRESS, --address ADDRESS
                        The ip address of the server
  -p [PORT [PORT ...]], --port [PORT [PORT ...]]
                        List of ports that must be equal to number of
virtual servers
  -r RESUME, --resume RESUME
                        Flag that tells the client whether to resume the
existing download in progress
```

```
$ ./client.py -o . -a 127.0.0.1 -i 1 -r -p 10001 10002 10003 10004
```

```
$ python client.py --output . --address 127.0.0.1 --interval 1 --resume --
port 10001 10002 10003 10004
```