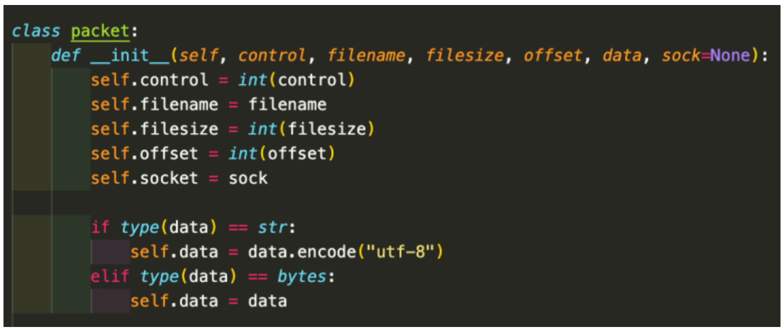
Adrian Bedford, 22973676
Oliver Lynch, 22989775

Networks Report

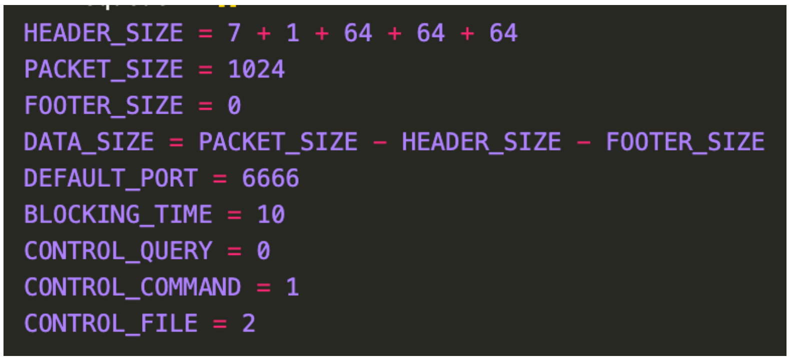
Protocol: All of our networking protocols are defined in a separate file to rake-p and rakeserver, titled networkstuff.py. The server was written in python.

We have defined a packet structure, where the first 7 bits are arbitrarily fixed at \x01 to identify the start of a packet. The 8th bit is a control bit and determines the packet type: 0 for system queries, 1 for command, 2 for file transfer and 3 for command return. 64 bits are reserved for the filename, 64 for the file size, 64 as an offset and the remaining bits for the file contents itself for a total of 1024 bits, but would expand to fill a larger size.



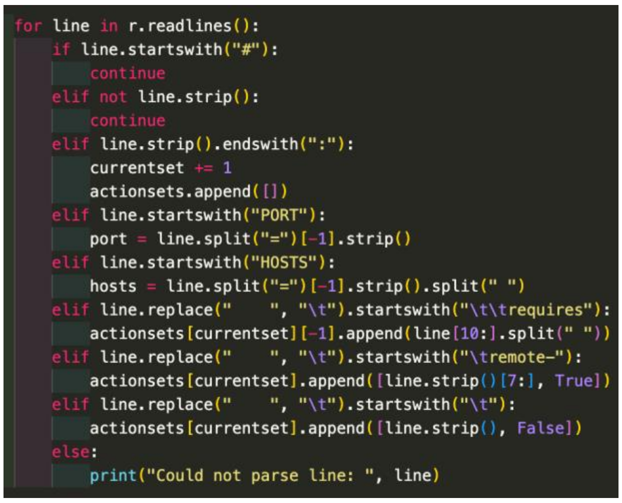
```
class packet:
    def __init__(self, control, filename, filesize, offset, data, sock=None):
        self.control = int(control)
        self.filename = filename
        self.filesize = int(filesize)
        self.offset = int(offset)
        self.socket = sock

        if type(data) == str:
            self.data = data.encode("utf-8")
        elif type(data) == bytes:
            self.data = data
```



```
HEADER_SIZE = 7 + 1 + 64 + 64 + 64
PACKET_SIZE = 1024
FOOTER_SIZE = 0
DATA_SIZE = PACKET_SIZE - HEADER_SIZE - FOOTER_SIZE
DEFAULT_PORT = 6666
BLOCKING_TIME = 10
CONTROL_QUERY = 0
CONTROL_COMMAND = 1
CONTROL_FILE = 2
```

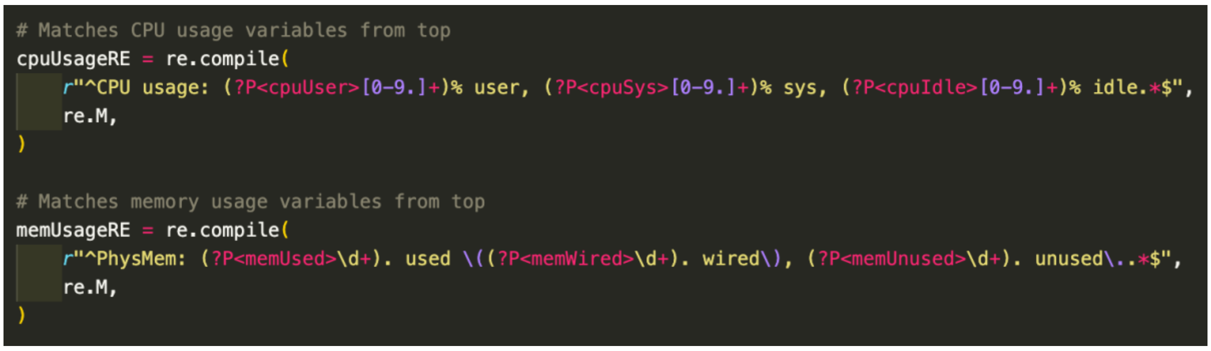
Before connecting to the server, the rakefile is read by the client and separated through a series of conditionals, where each line is evaluated by its starting and ending characters to assign things like the port, host, local and remote commands to respective variables.



```
for line in r.readlines():
    if line.startswith("#"):
        continue
    elif not line.strip():
        continue
    elif line.strip().endswith(":"):
        currentset += 1
        actionsets.append([])
    elif line.startswith("PORT"):
        port = line.split("=")[-1].strip()
    elif line.startswith("HOSTS"):
        hosts = line.split("=")[-1].strip().split(" ")
    elif line.replace("    ", "\t").startswith("\t\trequires"):
        actionsets[currentset][-1].append(line[10:].split(" "))
    elif line.replace("    ", "\t").startswith("\tremote-"):
        actionsets[currentset].append([line.strip()[7:], True])
    elif line.replace("    ", "\t").startswith("\t"):
        actionsets[currentset].append([line.strip(), False])
    else:
        print("Could not parse line: ", line)
```

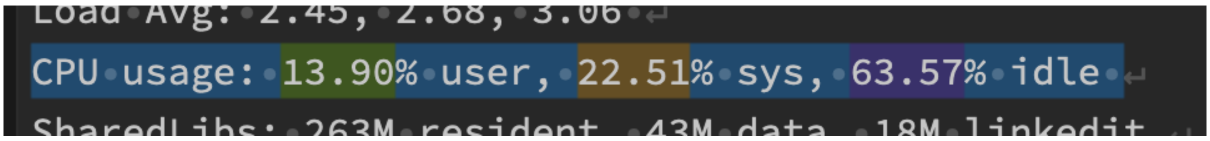
Adrian Bedford, 22973676
Oliver Lynch, 22989775

Once the file is parsed, the client attempts to connect to every server specified in the rakefile. To be able to send and receive the packets, we implemented a buffering system to interpret the bytestream into usable sequences of data. The servers are requested to bid based on their system usage to determine which is best to be used to execute commands. The bid is based on the current CPU and memory utilisation, averaged. Regex is used to interpret the output of top. The server with the lowest average is picked for execution.



```
# Matches CPU usage variables from top
cpuUsageRE = re.compile(
    r"^CPU usage: (?P<cpuUser>[0-9.]+)% user, (?P<cpuSys>[0-9.]+)% sys, (?P<cpuIdle>[0-9.]+)% idle.*$",
    re.M,
)

# Matches memory usage variables from top
memUsageRE = re.compile(
    r"^PhysMem: (?P<memUsed>\d+). used \((?P<memWired>\d+). wired\), (?P<memUnused>\d+). unused\..*$",
    re.M,
)
```



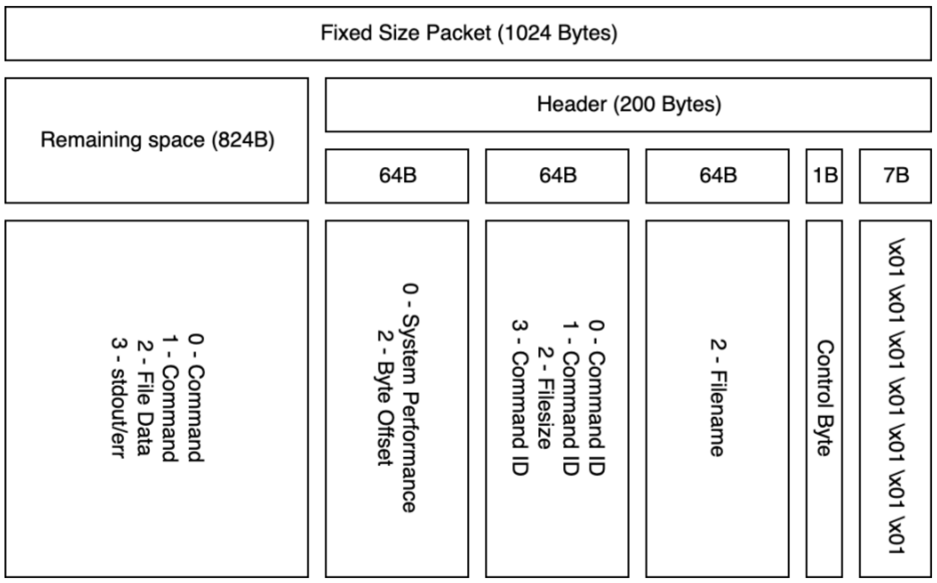
```
Load Avg: 2.45, 2.68, 3.06
CPU usage: 13.90% user, 22.51% sys, 63.57% idle
SharedLibs: 263M resident, 43M data, 18M linkedit
```

If the command is specified to run locally, it runs the commands as subprocesses. If remotely, the command is sent after the bidding process by the remoteProcess class from the networking file. Any required files are also sent to the server prior to the command execution request, ensuring the server has the necessary files to complete the command. The client and server both regularly check run poll to detect if any process is finished via acknowledgements from the server and performs all necessary reads and writes to the network. The poll function checks if any subprocesses are complete and also uses select() to determine if any sockets need to be read into the buffer or written to.

Although we did not implement a C version of the client, our network protocol would be relatively simple to implement as it does not use any python specific functionality, thus being agnostic to either language.

Adrian Bedford, 22973676
Oliver Lynch, 22989775

Here is a diagram of the packet itself:



The control byte determines which fields are used for what purpose. The filename field is interpreted as a utf-8 encoded bytestream, and the following two fields are interpreted as little-endian integers. The data field is always interpreted as a bytestream and may be decoded from utf-8 when necessary.

Compilation and Linking: Though we did not get the time to implement compilation and linking of multiple-file programs, we will outline the process we would have taken below.

All files listed as required in the rakefile would be sent to the server, and our program should already be capable of running a cmake command through the shell. The only difference between being able to run single-file execution set and multi-file one is that we haven't implemented the ability for our server to return a file to the client, it can only receive one.

Compilation and linking instead of local execution would run better if you were trying to locally execute on a low-spec laptop. You could outsource execution to better hardware and only require network access for results.