Q2. Clustershell

Run

Server

```
make server
./server.out <SERVER_PORT>
```

A server will be initialised on the local IP and port `SERVER_PORT`. If the port is busy, an error will be thrown.

Client

```
make client
./client.out <SERVER_IP> <SERVER_PORT> <CLIENT_PORT>
```

- `SERVER_IP`: IP address of the server
- `SERVER_PORT`: Port name of the server
- `CLIENT_PORT`: Port of which client should establish its own server to accept commands and run locally (see below for explanation)

Design

Server

- The clustershell server establishes a server on the given port and waits for client connections. On receiving a connection request, the server creates a new thread for each client. We have chosen threads instead of processes as the clients and the main process have to share some data. When a new request comes, the server checks the IP in config file and gets the machine name. If the machine name is not found, the connection is closed. On successful connection and teardown, the client thread informs the parent about the connection establishment/teardown and parent uses this information to keep track of active connections.

- When the server first establishes a connection with a client, the first message it expects is the client port on which the clustershell client is running it's own server to accept commands.

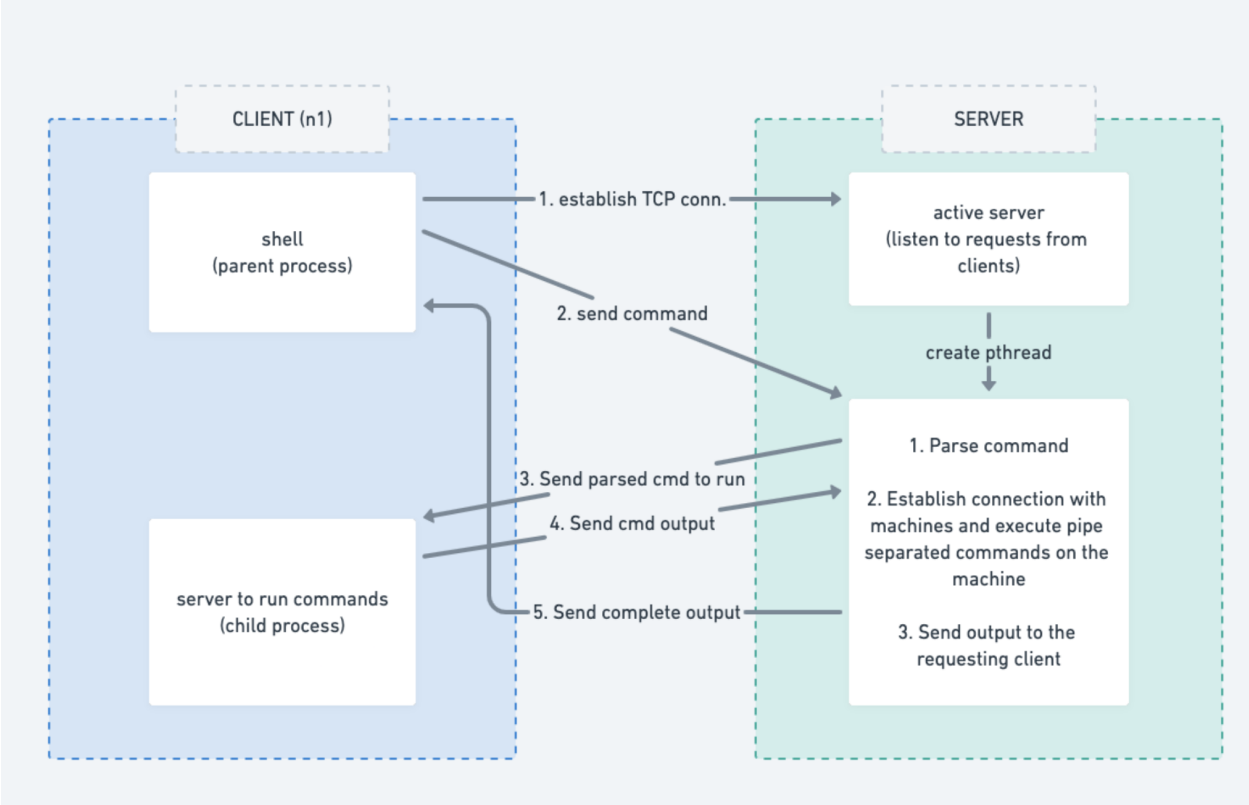
- When the server received a command from the client, it parses the pipe-separated commands and runs each command on the specific machine (or sequentially runs of each active connection in case of n* and gathers output). This is done by establishing a TCP connection with the IP given in config file and port given by client port. Piping is performed through sending the output of previous command along with the actual command to the client.

Client

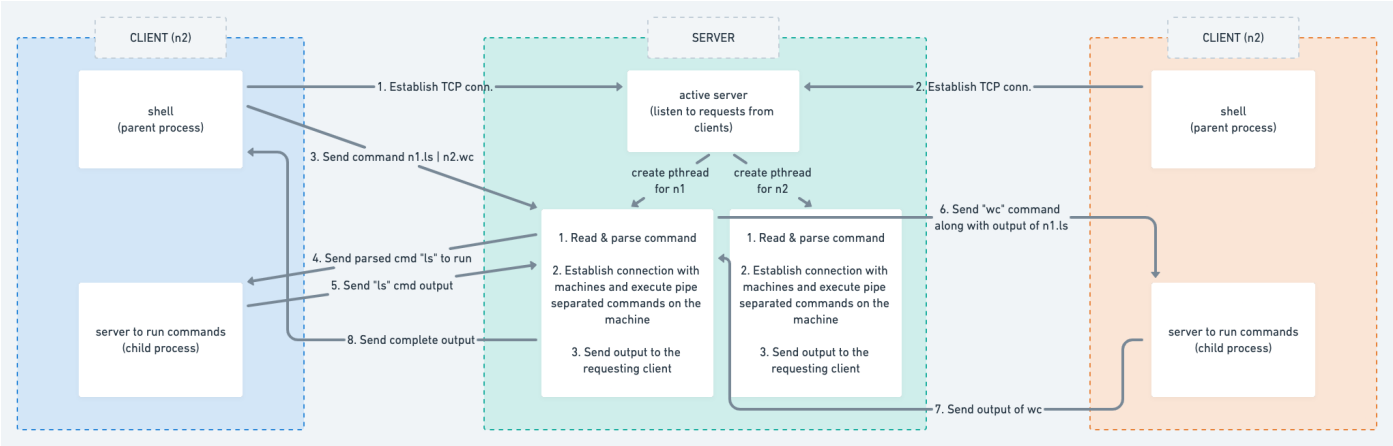
- Each client runs two processes. The main process runs the user-facing shell and acts as a networking client connected with the clustershell server. Upon receiving commands from the user through stdin, the main process sends the command to the server.
- The second process establishes it's own server on CLIENT_PORT and listens to requests from clustershell server to run commands on the machine and return the output.

Server <–> Client

The figure below shows an overview or server - client communication as explained above.



Example: n1.ls | n2.wc



View image in browser

0. **Server initialised:** When the server is run, it binds to the port given by command line argument `SERVER_PORT` and listens to client requests.
1. **First machine connects with the server:** When the first machine establishes a TCP connection with the server, the server creates a new thread for handling requests from this machine. The server first verifies that the IP exists in config and gets the machine name (n1). Then, the server expects the client to send the client port before sending any shell commands.
   Additionally, the client on initialising forks a child process that binds to a port and listens to requests from the server.
2. **Second machine connects with server:** Same as step 2.
3. **n1 sends `n1.ls | n2.wc` command to server:** The server parses the command and creates a linked list of pipe separated commands. Further, the server performs the below steps.
4. **Server sends command ls to n1:** The server establishes a TCP connection with n1 and sends the `ls` command to n1 and waits for the output. Note that here the clustershell server actually acts like a client.
5. **n1 responds back with output:** n1 sends the output of ls to the server.
6. **Server sends command wc to n2:** The server establishes a TCP connection with n2 and sends the wc command along with output of n1.ls to n2 and waits for the output. Note that here the clustershell server actually acts like a client.
7. **n2 responds back with output:** n2 sends the output of `n1.ls | n2.wc` to server.
8. **Server sends final output to n1:** The server sends the final output to requesting client n1.

Features

- The server keeps track of open connections that can be queried by a client using `nodes` command.
- The server accepts connections only from clients specified in the config file.
- There can be more than one client running on the same IP. The command line argument `CLIENT_PORT` is used to differentiate between such clients.
- The client can run commands locally (no machine specified, eg: `ls`), or on a particular machine (eg: `n2.ls`), or on all current active connections (eg: `n*.ls`)
- The shell is able to execute piped commands (eg: `n2.ls | n1.wc`)
- The shell supports the cd command
- To exit server, press `Ctrl+C`
- To exit shell on client, run `exit`

Examples

```
ls
```

```
n2.ls | n1.wc
```

```
n*.cd ../
```

```
n1.cat file | n2.sort | n3.uniq
```

```
nodes
```

```
n*.ls | n2.wc
```

Screenshots

Due to multiple terminals, the text might not be visible in this doc. Please zoom in or refer to the link attached with each image to view it in the browser.

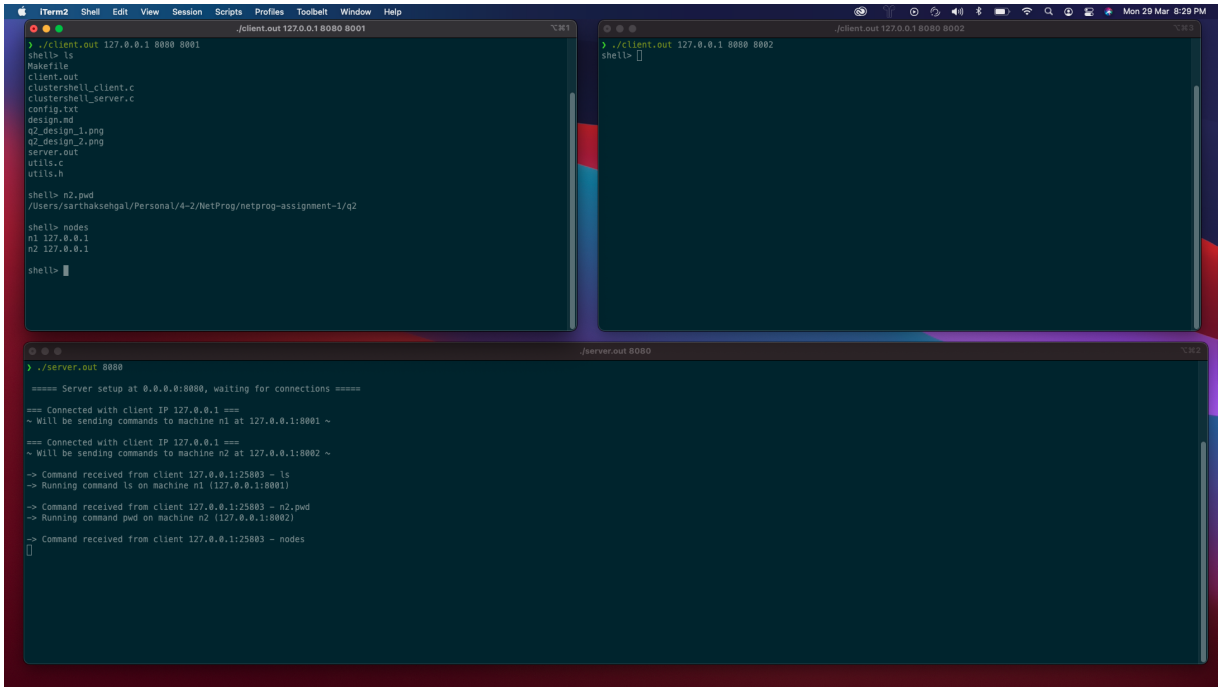


Image 1

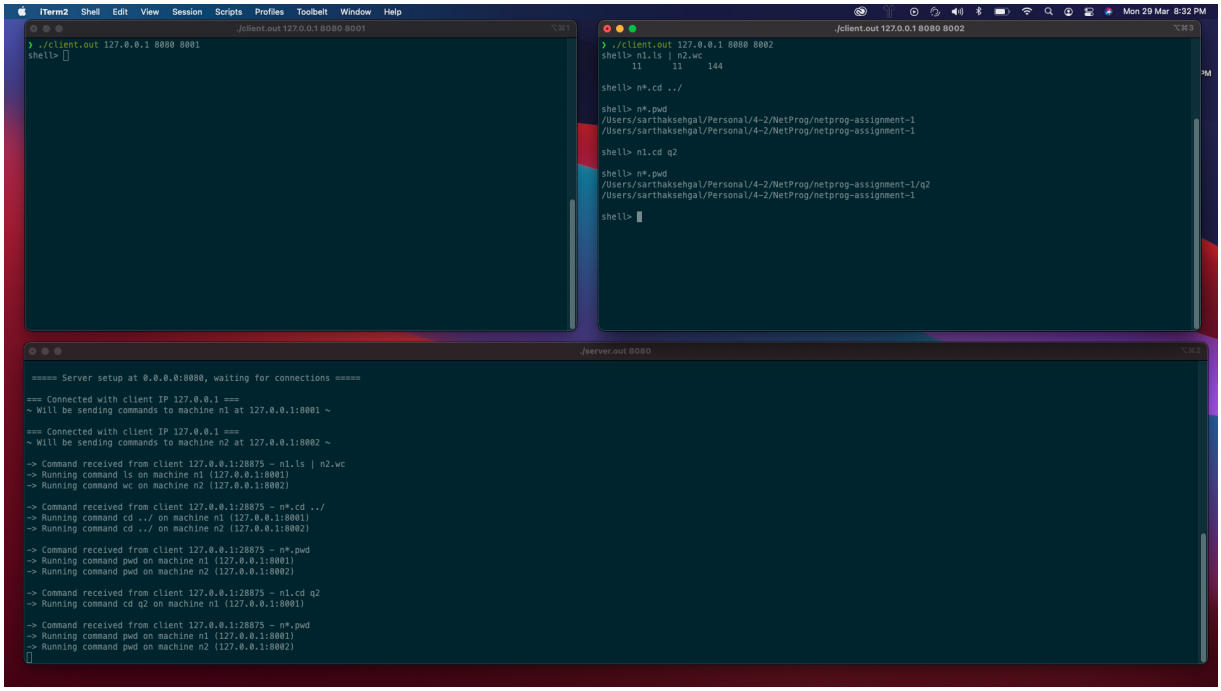
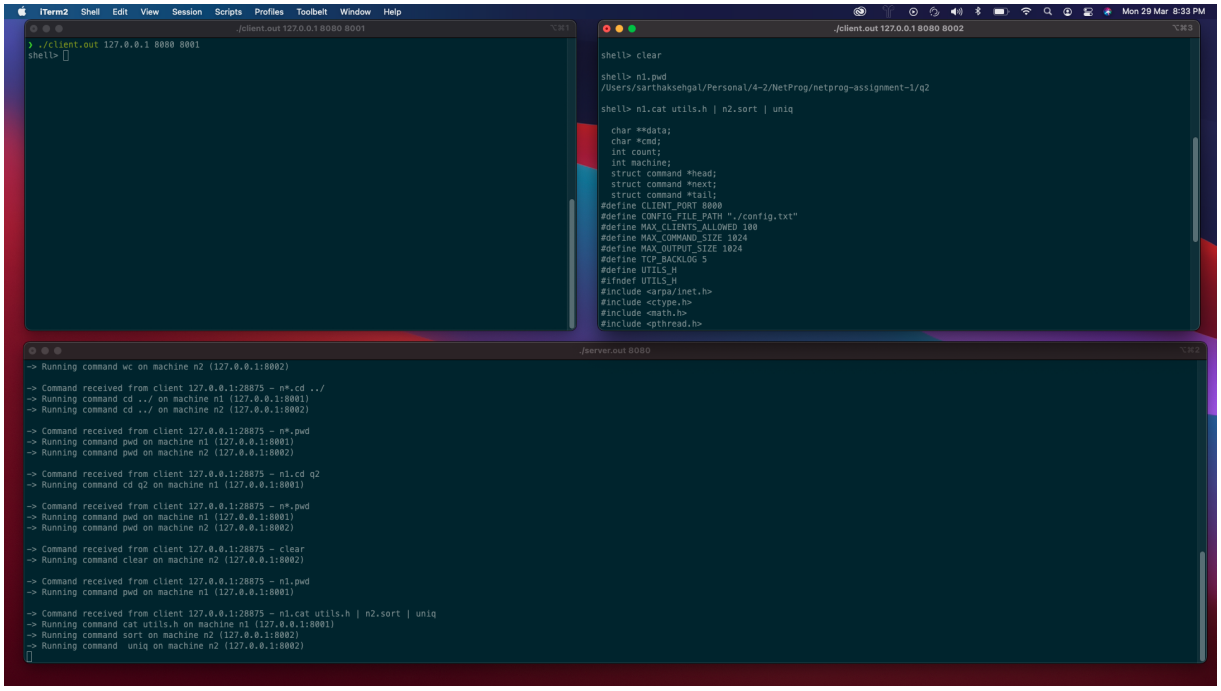
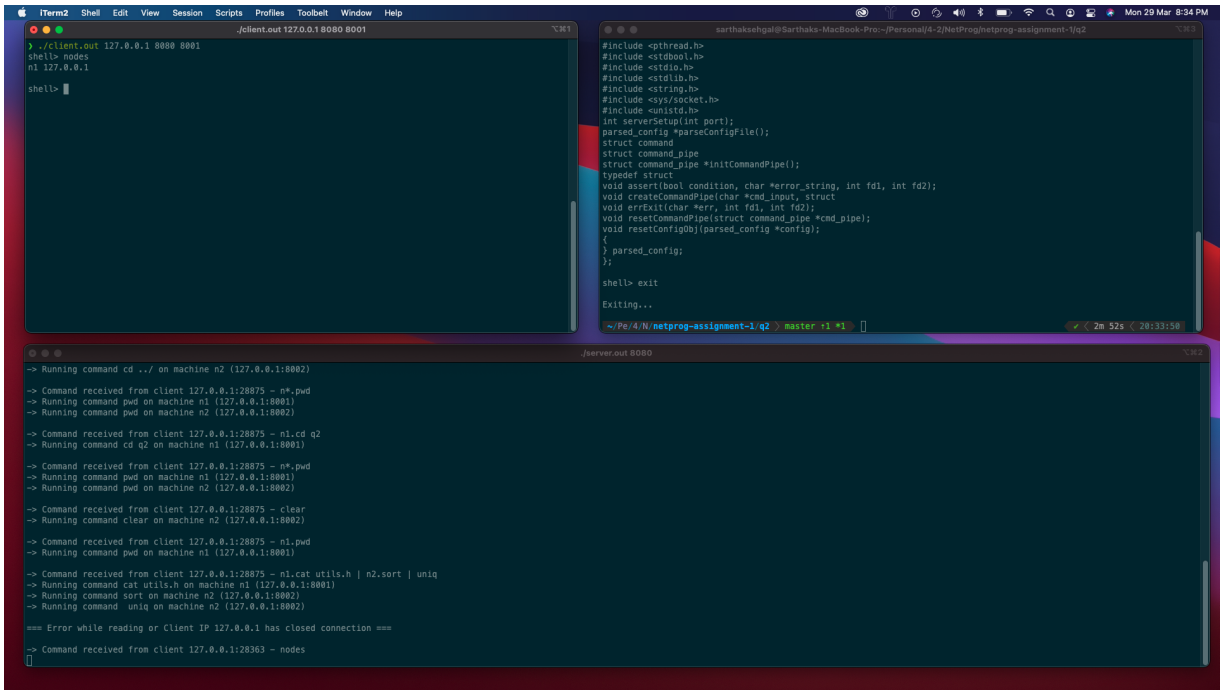


Image 2



[Image 3]



[Image 4]

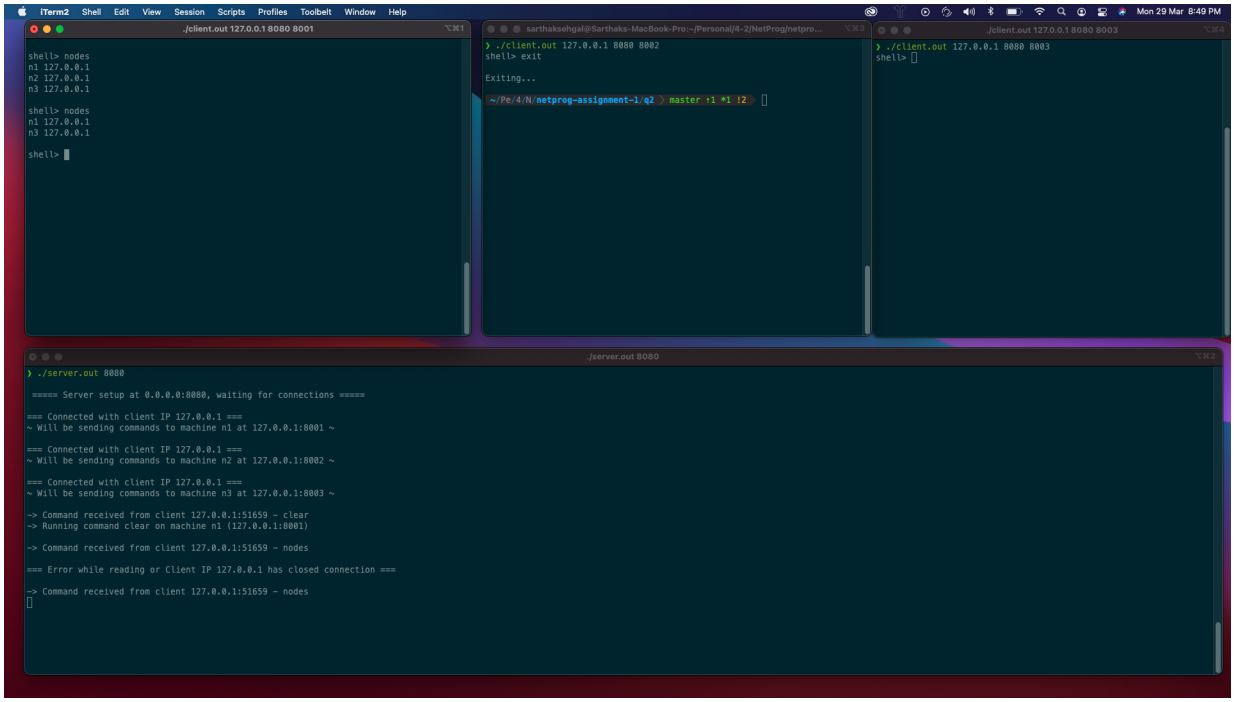

iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
Mon 29 Mar 8:49 PM
./client.out 127.0.0.1 8080 8001
shell> nodes
n1 127.0.0.1
n2 127.0.0.1
n3 127.0.0.1
shell> nodes
n1 127.0.0.1
n3 127.0.0.1
shell>
./client.out 127.0.0.1 8080 8002
shell> exit
Exiting...
~/Pe/4/N/netprog-assignment-1/q2 master ↑1 *1 !2
./client.out 127.0.0.1 8080 8003
shell>
./server.out 8080
===== Server setup at 0.0.0.0:8080, waiting for connections =====
=== Connected with client IP 127.0.0.1 ===
~ Will be sending commands to machine n1 at 127.0.0.1:8001 ~
=== Connected with client IP 127.0.0.1 ===
~ Will be sending commands to machine n2 at 127.0.0.1:8002 ~
=== Connected with client IP 127.0.0.1 ===
~ Will be sending commands to machine n3 at 127.0.0.1:8003 ~
-> Command received from client 127.0.0.1:51659 - clear
-> Running command clear on machine n1 (127.0.0.1:8001)
-> Command received from client 127.0.0.1:51659 - nodes
=== Error while reading or Client IP 127.0.0.1 has closed connection ===
-> Command received from client 127.0.0.1:51659 - nodes

Image 5