

DESIGN DOCUMENT

P2: Cluster Shell

Problem Statement - Create a basic cluster shell that can coordinate commands to be run on different nodes.

1. Can execute basic shell commands (ls, wc, cat etc.)
2. Supports the pipe operator between different nodes - | (n1.ls | n2.sort | n3. uniq)
3. Supports changing directories using cd command (n1.cd dir1)
4. Supports running commands on all machines and combining the output (n*.ls)

Usage

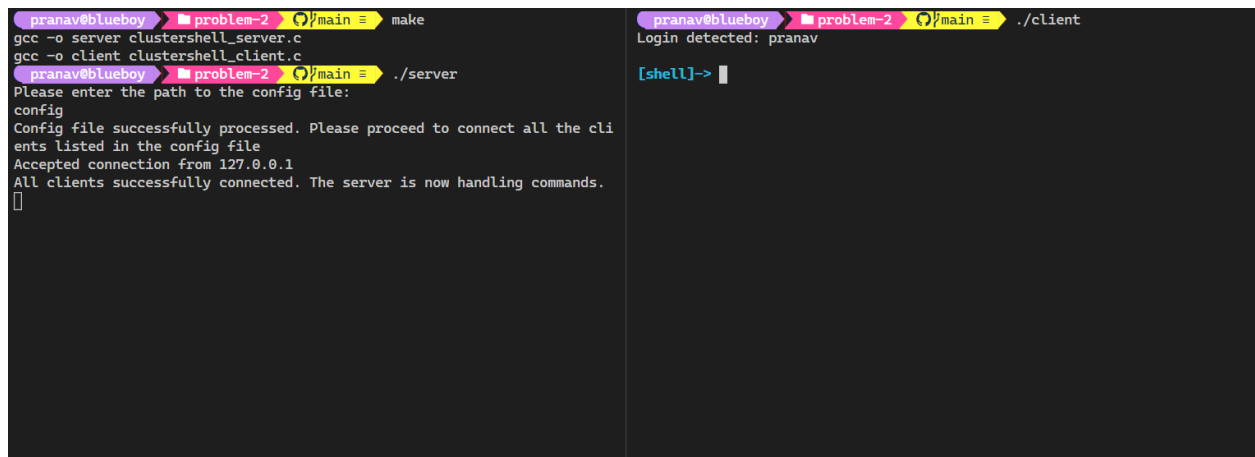
The server and client executables can be compiled using the Makefile:

On the server node:

```
make
./server
```

The server will ask for the path to the *config* file which needs to be entered by the user.

Example screenshot:



```
pranav@blueboy ~$ make
gcc -o server clustershell_server.c
gcc -o client clustershell_client.c
pranav@blueboy ~$ ./server
Please enter the path to the config file:
config
Config file successfully processed. Please proceed to connect all the clients listed in the config file
Accepted connection from 127.0.0.1
All clients successfully connected. The server is now handling commands.
[]

pranav@blueboy ~$ ./client
Login detected: pranav
[shell]-> 
```

On all the nodes listed in config file:

```
make
./client
```

Note: 1. The client and server can be compiled directly using gcc as well.

Note: 2. To cd into a directory append a "/" to it. Eg. cd Desktop/. instead of cd Desktop.

Messaging format between server and client:

Clients do not directly communicate, instead they are only connected to the server which is responsible for all the coordination. All communication between the clients and the server happens in a fixed messaging format.

The messaging format is as follows:

- *Command messages from shell to server: 6 character command header + command*
- *Command messages to from server to client: 6 character command header + 6 digit input header + input + command*
- *Output messages: 6 character header + output*

The first letter of the header is c/o/i signifying command/output/input

The next 5 characters specify the length of the corresponding command/output/input.

Eg. If the user types in n1.ls on the shell in n2, then it will be encoded as

"c00005n1.ls" and sent to the server. The server will send "c00002i00000ls" to n1 and get the output from n1 as "o00384<output of ls on n1>". This output will be sent to n2 where it will be decoded into <output of ls> and printed.

Server Design

1. Server is run on any one machine. All clients must specify the IP address of the

server as a preprocessing directive in the client code (SERV_ADDRESS)

2. In the beginning all the clients listed in the config file connect to the server.
3. Upon receiving a command from any client, it connects to the specified machines and sends the required commands and inputs to those commands to those nodes.
4. Returns final output to the client that initially sent the command.
5. If any command has to be run on all the nodes (n^*) then it is sent to all the nodes and the output is concatenated and used for further commands or sent to the client that sent the command.

Client Design

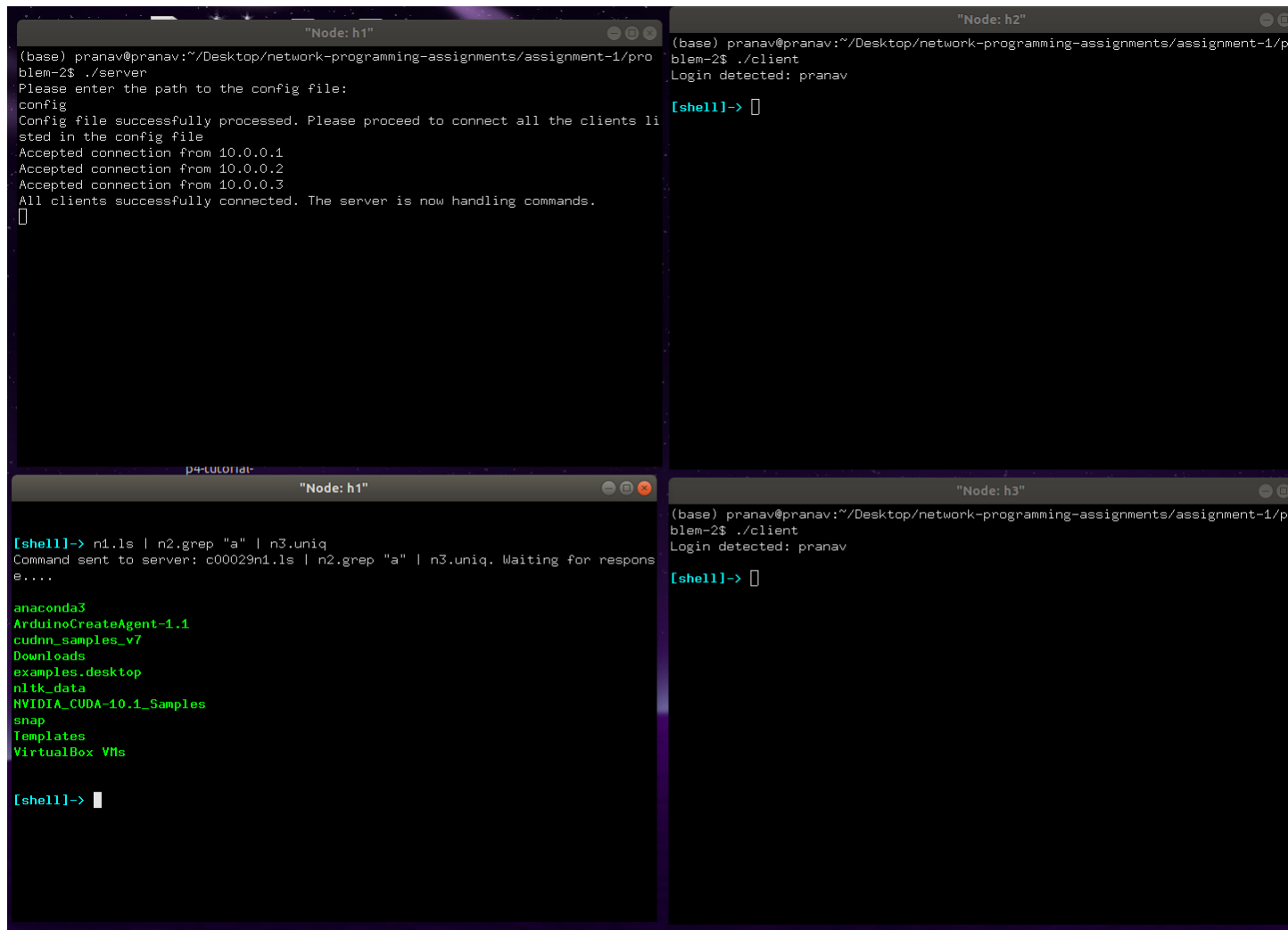
1. Client is run on all machines that are listed in the config file
2. Client connects to server on TCP, and the server IP is specified as a preprocessing directive in the client and server code (SERV_ADDRESS)
3. Client sends command to server, and server coordinates the connected clients to execute the command.
4. It runs a mini-server to which the server connects and sends commands that need to be executed on that particular node.
5. There are two processes: one which handles the shell with the client, and one which handles the incoming commands from the server for this client.
6. Each of the two processes has its own TCP connection to the server.

Assumptions:

1. All clients listed in the config file connect in the beginning itself and none of them leave before all commands are over.
2. There are no commands that require manual user input from the shell (stdin).
3. Commands, inputs and outputs are of maximum string length 99999 including all ending characters and newlines.
4. Nodes are named as $n_1, n_2, n_3, \dots, n_N$.
5. Nodes are listed in order in the config file and no node number is missing.

Screenshots:

For these screenshots, 3 hosts were run inside a network emulator called Mininet.



```
"Node: h1"
(base) pranav@pranav:~/Desktop/network-programming-assignments/assignment-1/problem-2$ ./server
Please enter the path to the config file:
config
Config file successfully processed. Please proceed to connect all the clients listed in the config file
Accepted connection from 10.0.0.1
Accepted connection from 10.0.0.2
Accepted connection from 10.0.0.3
All clients successfully connected. The server is now handling commands.
[]

"Node: h2"
(base) pranav@pranav:~/Desktop/network-programming-assignments/assignment-1/problem-2$ ./client
Login detected: pranav
[shell1]-> []

p4-tutorial-
"Node: h1"
[shell1]-> n1.ls | n2.grep "a" | n3.uniq
Command sent to server: c00029n1.ls | n2.grep "a" | n3.uniq. Waiting for response....
anaconda3
ArduinoCreateAgent-1.1
cudnn_samples_v7
Downloads
examples.desktop
nltk_data
NVIDIA_CUDA-10.1_Samples
snap
Templates
VirtualBox VMs
[shell1]-> []

"Node: h3"
(base) pranav@pranav:~/Desktop/network-programming-assignments/assignment-1/problem-2$ ./client
Login detected: pranav
[shell1]-> []
```

```
XTerm
Tue 23:45

"Node: h1"
(base) pranav@pranav:~/Desktop/network-programming-assignments/assignment-1/pro
blem-2$ ./server
Please enter the path to the config file:
config
Config file successfully processed. Please proceed to connect all the clients li
sted in the config file
Accepted connection from 10.0.0.1
Accepted connection from 10.0.0.2
Accepted connection from 10.0.0.3
All clients successfully connected. The server is now handling commands.
[]

"Node: h2"
[shell]-> n1.ls
Command sent to server: c00005n1.ls. Waiting for response....
anaconda3
ArduinoCreateAgent-1.1
cudnn_samples_v7
Desktop
Documents
Downloads
examples.desktop
hpl
hplresults-1
Music
new.py
nltk_data
NVIDIA_CUDA-10.1_Samples
Pictures
Public
snap
sophos
Templates
Videos
VirtualBox VMs

[shell]-> []

"Node: h1"
[shell]-> n1.ls | n2.grep "a" | n3.uniq
Command sent to server: c00029n1.ls | n2.grep "a" | n3.uniq. Waiting for respons
e....
anaconda3
ArduinoCreateAgent-1.1
cudnn_samples_v7
Downloads
examples.desktop
nltk_data
NVIDIA_CUDA-10.1_Samples
snap
templates
VirtualBox VMs

[shell]-> []

"Node: h3"
[shell]-> n2.ls
Command sent to server: c00005n2.ls. Waiting for response....
anaconda3
ArduinoCreateAgent-1.1
cudnn_samples_v7
Desktop
Documents
Downloads
examples.desktop
hpl
hplresults-1
Music
new.py
nltk_data
NVIDIA_CUDA-10.1_Samples
Pictures
Public
snap
sophos
Templates
Videos
VirtualBox VMs

[shell]-> []
```