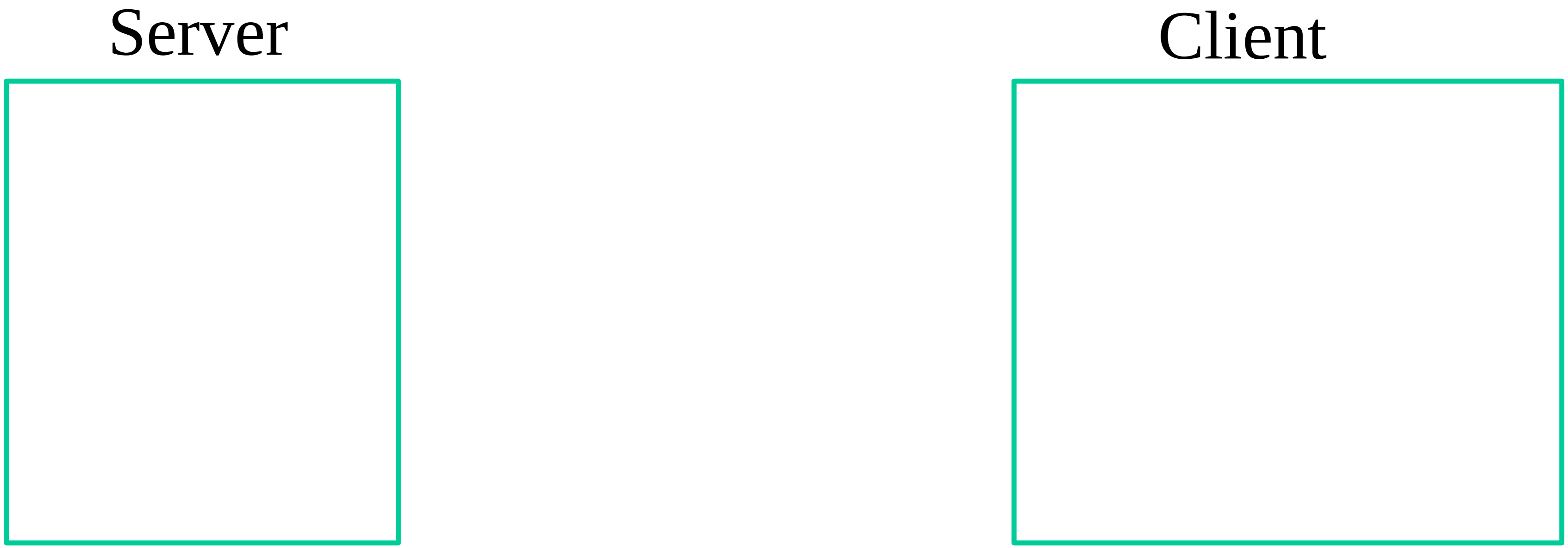
Networking Assignment - 2

Team Emertxe

WAP to implement the remote command execution using Linux environment

Assignment - 2

Let’s understand what is remote command execution?

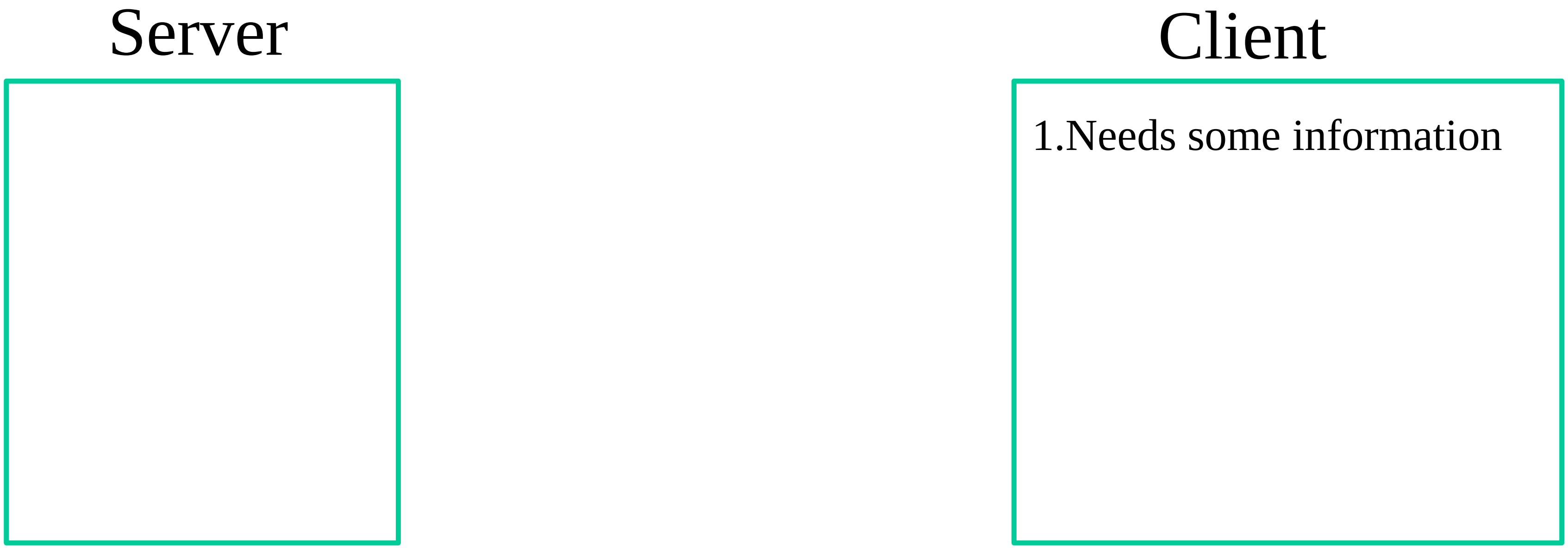


Server

Client

Assignment - 2

Let’s understand what is remote command execution?



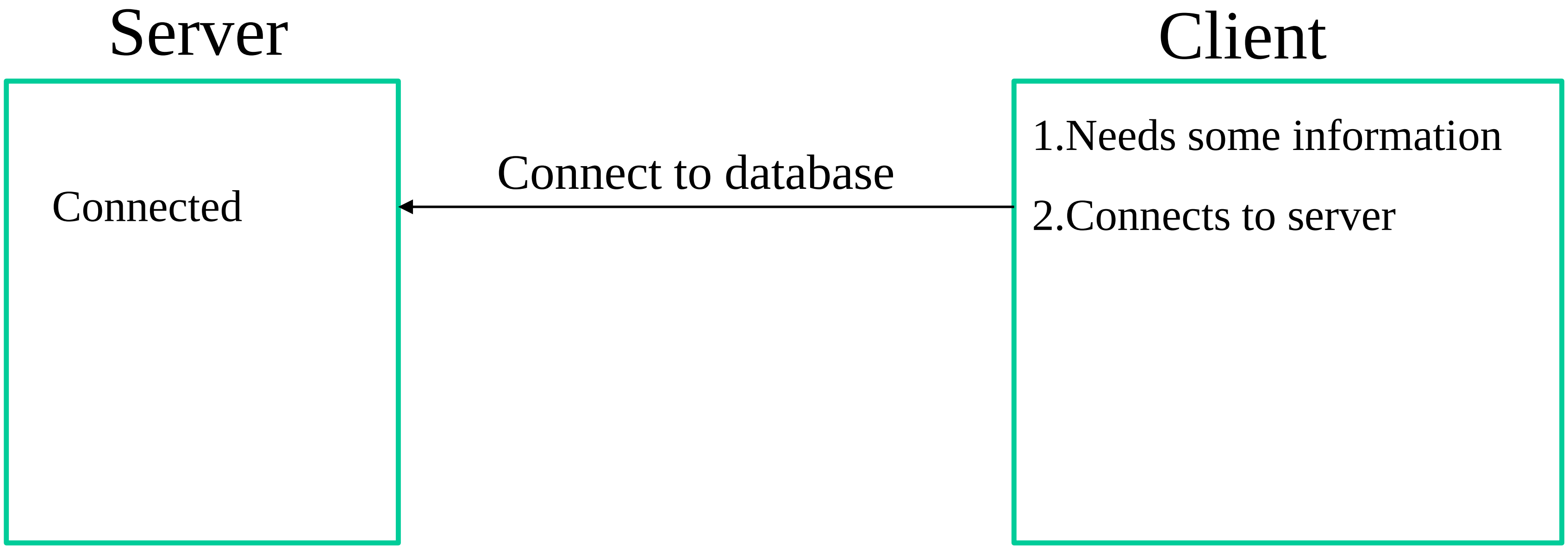
Server

Client

1.Needs some information

Assignment - 2

Let’s understand what is remote command execution?



Server

Connected

Connect to database

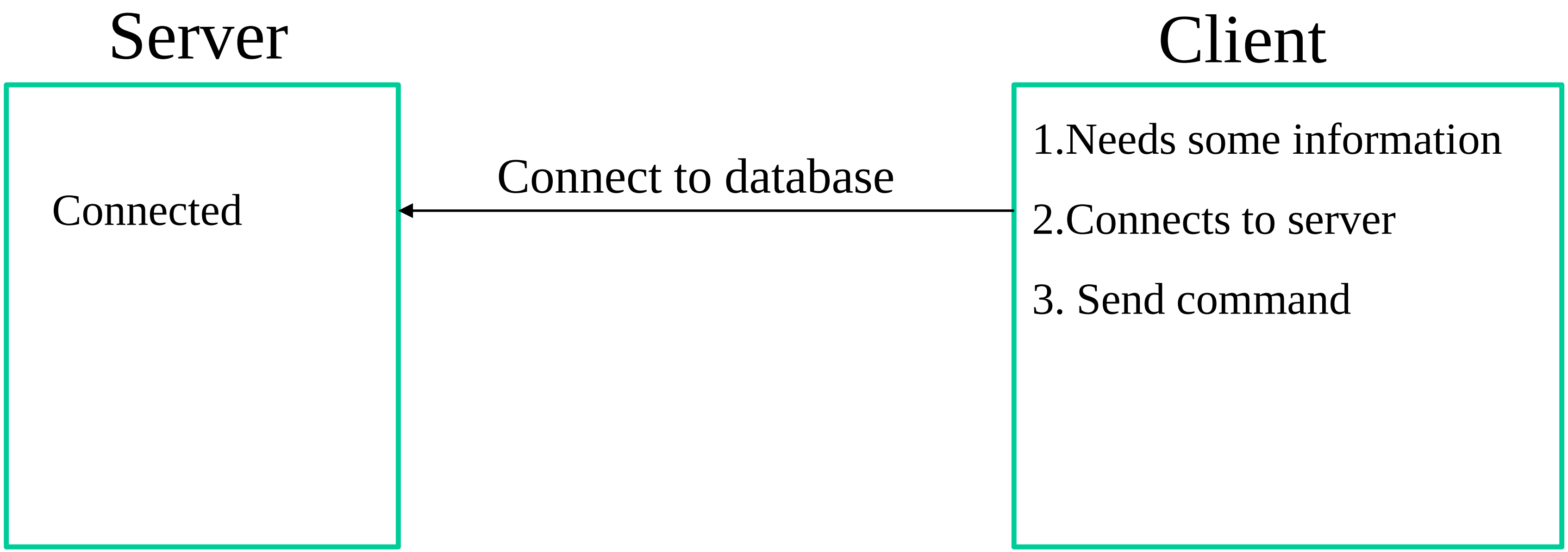
Client

1.Needs some information

2.Connects to server

Assignment - 2

Let’s understand what is remote command execution?



Server

Connected

Connect to database

Client

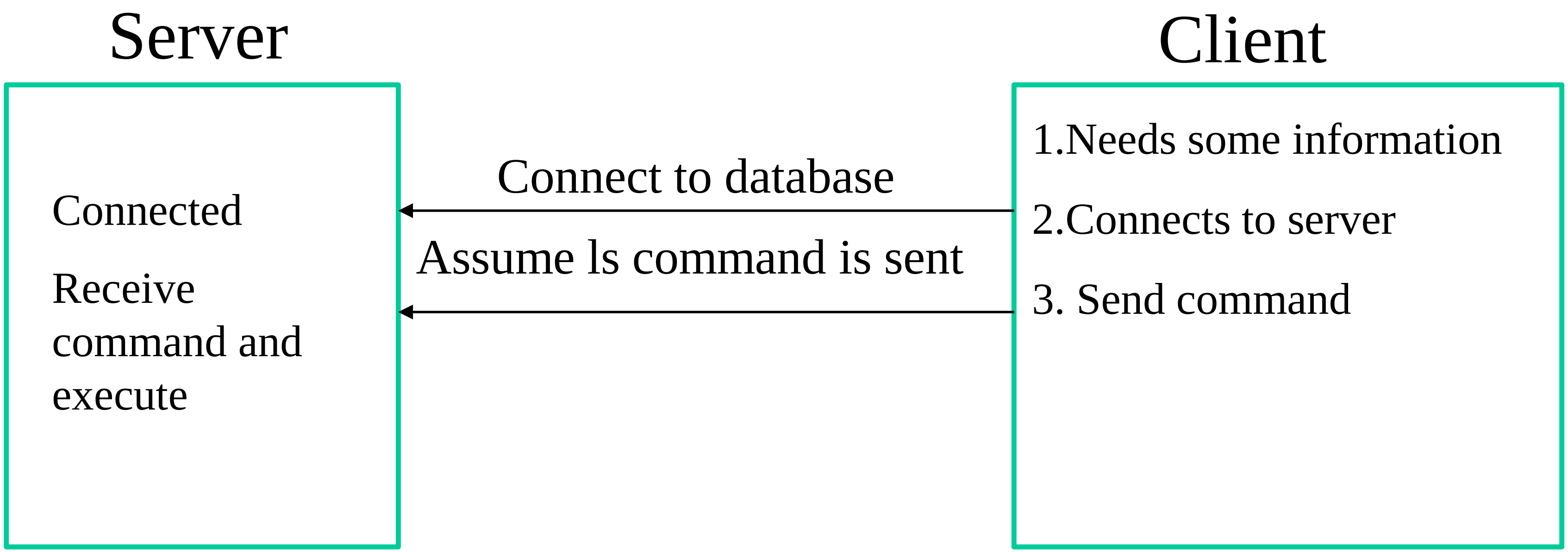
1.Needs some information

2.Connects to server

3. Send command

Assignment - 2

Let’s understand what is remote command execution?



Server

Connected

Receive command and execute

Connect to database

Assume ls command is sent

Client

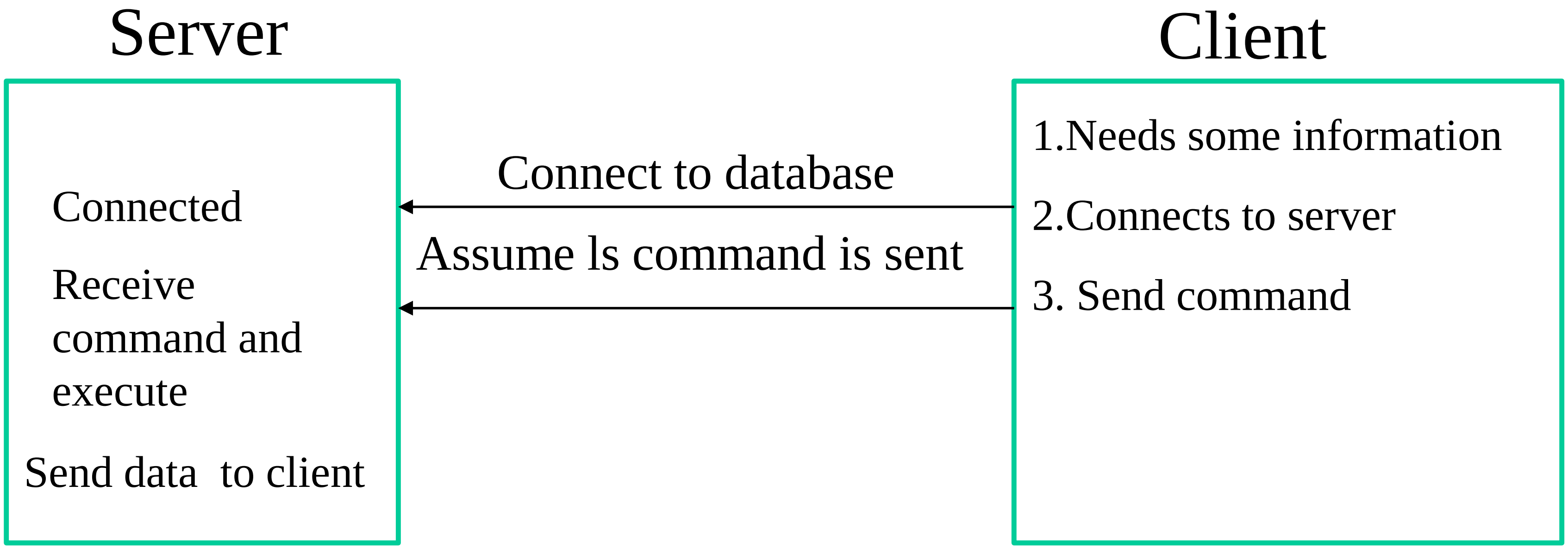
1.Needs some information

2.Connects to server

3. Send command

Assignment - 2

Let’s understand what is remote command execution?



Server

Client

1.Needs some information

2.Connects to server

3. Send command

Connect to database

Assume ls command is sent

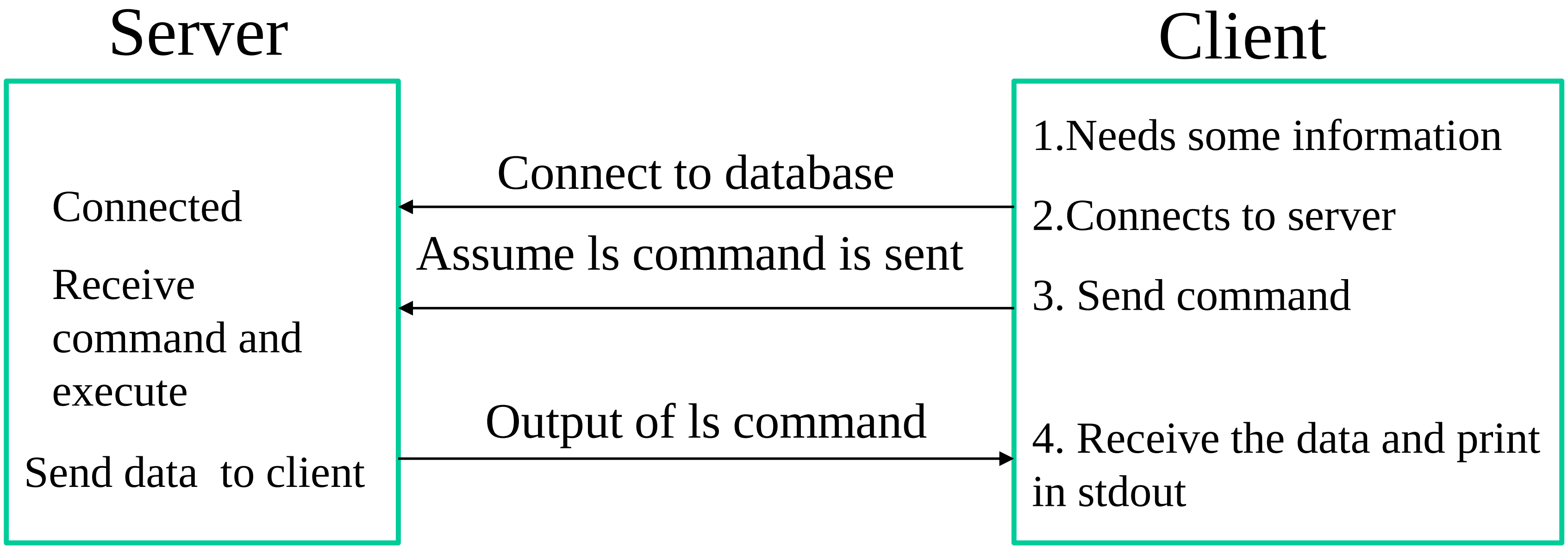
Connected

Receive command and execute

Send data to client

Assignment - 2

Let’s understand what is remote command execution?



Server

Client

1.Needs some information

2.Connects to server

3. Send command

4. Receive the data and print in stdout

Connect to database

Assume ls command is sent

Output of ls command

Connected

Receive command and execute

Send data to client

Assignment - 2

Let see the requirements of this assignment.

Assignment - 2

Client:

Assignment - 2

Client:

✓ It has to send any standard Linux Shell command to the server using ‘command request’ packet along with no. of times the command need to be executed.

Assignment - 2

Client:

- ✓ It has to send any standard Linux Shell command to the server using 'command request' packet along with no. of times the command need to be executed.

Server:

- ✓ It will parse the command then execute it the number of times specified.

Assignment - 2

Client:

- ✓ It would send any standard Linux Shell command to the server using ‘command request’ packet along with no. of times the command need to be executed.

Server:

- ✓ It will parse the command then execute it the number of times specified.
- ✓ Apart from printing the output, the server should capture the output in a file.

Assignment - 2

Client:

✓ It would send any standard Linux Shell command to the server using ‘command request’ packet along with no. of times the command need to be executed.

Server:

✓ It would parse the command then execute it the number of times specified.

✓ Apart from printing the output, the server should capture the output in a file.

✓ This file would be then read by the server and the output would be sent back to the client in terms of 64 bytes of ‘data’ packets along with the packet number.

Assignment - 2

Client:

- ✓ It would send any standard Linux Shell command to the server using ‘command request’ packet along with no.of times the command need to be executed.

Server:

- ✓ It would parse the command then execute it the number of times specified.
- ✓ Apart from executing it, the server would capture the output in a file.
- ✓ This file would be then read by the server and the output would be sent back to the client in terms of 64 bytes of ‘data’ packets along with the packet number.
- ✓ If the data size is less then 64 bytes it needs to be specified by the server.

Assignment - 2

Client:

- ✓ It would send any standard Linux Shell command to the server using ‘command request’ packet along with no.of times the command need to be executed.
- ✓ As this connection is unreliable UDP connection the client should acknowledge back with and ACK packet along with the packet number.

Server:

- ✓ It would parse the command then execute it the number of times specified.
- ✓ Apart from executing it, the server would capture the output in a file.
- ✓ This file would be then read by the server and the output would be sent back to the client in terms of 64 bytes of ‘data’ packets along with the packet number.
- ✓ If the data size is less then 64 bytes it needs to be specified by the server.

Assignment - 2

Client:

- ✓ It would send any standard Linux Shell command to the server using 'command request' packet along with no.of times the command need to be executed.
- ✓ As this connection is unreliable UDP connection the client should acknowledge back with and ACK packet along with the packet number.

Server:

- ✓ The server would send the next packet only after receiving the successful ACK for the previous packet.

Assignment - 2

Packets need to use:

Assignment - 2

Packets need to use:

1. Command Request packet.

Assignment - 2

Packets need to use:

1. Command Request packet.

Command	No.of times

Assignment - 2

Packets need to use:

1. Command Request packet.

Command	No.of times

2. Data packet:

Assignment - 2

Packets need to use:

1. Command Request packet.

Command	No.of times

2. Data packet:

Data (64 B)	Pack_no	N_flag

Assignment - 2

Packets need to use:

1. Command Request packet.

Command	**No.of times**

2. Data packet:

Data (64 B)	**Pack_no**	**N_flag**

3. Acknowledgement packet:

Assignment - 2

Packets need to use:

1. Command Request packet.

Command	**No.of times**

2. Data packet:

Data (64 B)	**Pack_no**	**N_flag**

3. Acknowledgement packet:

Data (64 B)	**Pack_no**

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)

    [123423]

$ ./udp_client

Enter any standard LS command :  date
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :  date
Enter no.of times to be executed :
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :  date
Enter no.of times to be executed :  3
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :  date
Enter no.of times to be executed :  3
```

Output.txt

```
Wed Apr 28 13:27:19 IST 2020
Wed Apr 28 13:27:19 IST 2020
Wed Apr 28 13:27:19 IST 2020
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :  date
Enter no.of times to be executed :  3
Wed Apr 28 13:27:19 IST 2020
Wed Apr 28 13:27:19 IST 2020
Wed Apr
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :  date
Enter no.of times to be executed :  3
Wed Apr 28 13:27:19 IST 2020
Wed Apr 28 13:27:19 IST 2020
Wed Apr (64 bytes of 1th packet received from server, sending ack)
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)

    [123423]

$ ./udp_client

Enter any standard LS command :  date

Enter no.of times to be executed :  3

Wed Apr 28 13:27:19 IST 2020

Wed Apr 28 13:27:19 IST 2020

Wed Apr (64 bytes of 1th packet received from server, sending ack)

28 13:27:19 IST 2020
```

Assignment - 2

Sample execution:

```
$ ./udp_server & (Running server in background)
    [123423]
$ ./udp_client
Enter any standard LS command :  date
Enter no.of times to be executed :  3
Wed Apr 28 13:27:19 IST 2020
Wed Apr 28 13:27:19 IST 2020
Wed Apr (64 bytes of 1th packet received from server, sending ack)
28 13:27:19 IST 2020 (20 bytes of 2th packet received from server,sending ack)
```

Assignment - 2

Implementation details:

1. Separate command request, data and ACK packets needs to be defined.
2. Output of each command should be written in a Separate file both in the client side and server side.
3. Use C file operations
4. Use UDP sockets

Assignment - 2

Pre-requisites:

- ✓ Knowledge about UDP sockets, How to read and send packages.
- ✓ Good knowledge about FILE I/O’s.
- ✓ Working dup and exec system calls.

Objective

- ✓ To understand how to use UDP sockets.