

Assignment 8

Implement a Peer-to-Peer Chat Application

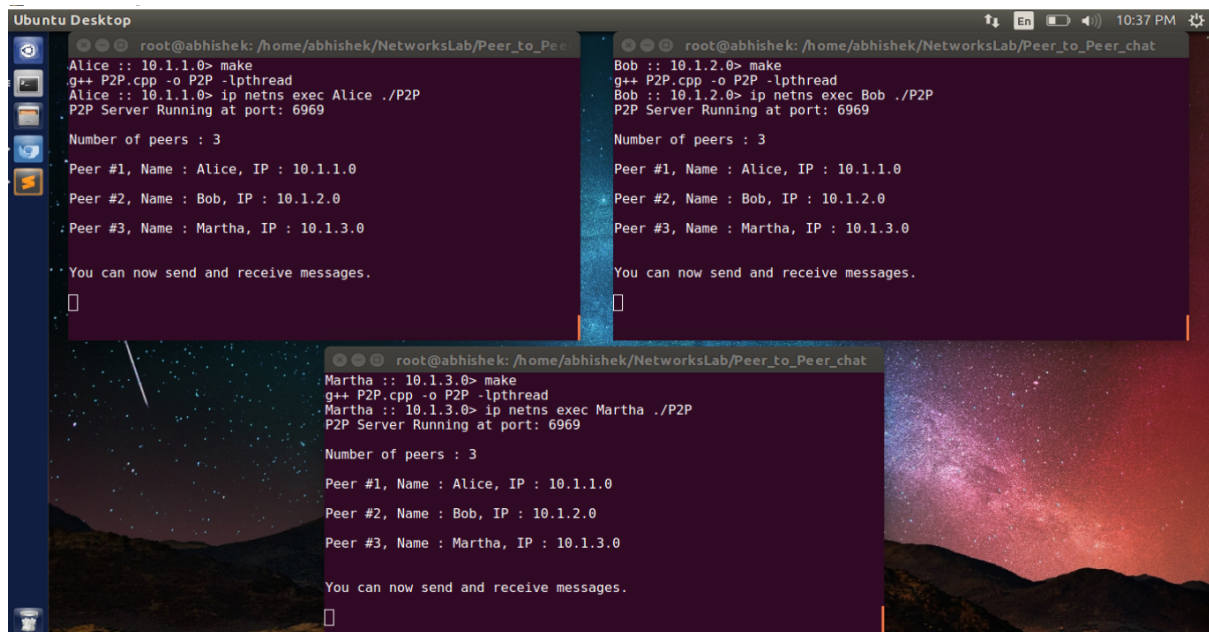
Documentation

Rahul Aditya (18CS30032)

Abhishek Srivastava (18CS10068)

For input output testing, we created 3 network namespaces Alice, Bob and Martha with IP addresses 10.1.1.0, 10.1.2.0 and 10.1.3.0 respectively. Then we ran a copy of the process on each of these network namespaces using `ip netns exec <namespace> ./P2P`

Sample Input Output :



```
root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Alice :: 10.1.1.0> make
g++ P2P.cpp -o P2P -lpthread
Alice :: 10.1.1.0> ip netns exec Alice ./P2P
P2P Server Running at port: 6969

Number of peers : 3
Peer #1, Name : Alice, IP : 10.1.1.0
Peer #2, Name : Bob, IP : 10.1.2.0
Peer #3, Name : Martha, IP : 10.1.3.0

You can now send and receive messages.
[]

root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Bob :: 10.1.2.0> make
g++ P2P.cpp -o P2P -lpthread
Bob :: 10.1.2.0> ip netns exec Bob ./P2P
P2P Server Running at port: 6969

Number of peers : 3
Peer #1, Name : Alice, IP : 10.1.1.0
Peer #2, Name : Bob, IP : 10.1.2.0
Peer #3, Name : Martha, IP : 10.1.3.0

You can now send and receive messages.
[]

root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Martha :: 10.1.3.0> make
g++ P2P.cpp -o P2P -lpthread
Martha :: 10.1.3.0> ip netns exec Martha ./P2P
P2P Server Running at port: 6969

Number of peers : 3
Peer #1, Name : Alice, IP : 10.1.1.0
Peer #2, Name : Bob, IP : 10.1.2.0
Peer #3, Name : Martha, IP : 10.1.3.0

You can now send and receive messages.
[]
```

All the namespaces are running a copy of the process. Each of them creates an array containing information about the peer group, by accessing the `peer_details.txt` file. Now they are ready to send and receive messages.

The **peer_details.txt** file has the following format:

```
<number of peers>
<peer name>
<peer ip>
```

Eg.

```
3
Alice
10.1.1.0
Bob
10.1.2.0
Martha
10.1.3.0
```

```
root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer
Peer #1, Name : Alice, IP : 10.1.1.0
Peer #2, Name : Bob, IP : 10.1.2.0
Peer #3, Name : Martha, IP : 10.1.3.0
You can now send and receive messages.
Bob/Hello to Bob from Alice
Martha/Hello to Martha from Alice
Bob : Hello to Alice from Bob
Martha : Hello to Alice from Martha
Connection with Bob timed out
Connection with Martha timed out

root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Peer #1, Name : Alice, IP : 10.1.1.0
Peer #2, Name : Bob, IP : 10.1.2.0
Peer #3, Name : Martha, IP : 10.1.3.0
You can now send and receive messages.
Alice : Hello to Bob from Alice
Alice/Hello to Alice from Bob
Martha/Hello to Martha from Bob
Martha : Hello to Bob from Martha
Connection with Alice timed out
Connection with Martha timed out

root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Peer #1, Name : Alice, IP : 10.1.1.0
Peer #2, Name : Bob, IP : 10.1.2.0
Peer #3, Name : Martha, IP : 10.1.3.0
You can now send and receive messages.
Alice : Hello to Martha from Alice
Bob : Hello to Martha from Bob
Alice/Hello to Alice from Martha
Bob/Hello to Bob from Martha
Connection with Alice timed out
Connection with Bob timed out
```

First Alice sends **Bob/Hello to Bob from Alice** and **Martha/Hello to Martha from Alice**
Next Bob sends **Alice/Hello to Alice from Bob** and **Martha/Hello to Martha from Bob**
Finally Martha sends **Alice/Hello to Alice from Martha** and **Bob/Hello to Bob from Martha**

All the messages are received by the peers they are sent to.

Finally after 120s all the peers timeout from the rest due to inactivity. Connection timeout messages are displayed.

```
root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_ch
Peer #3, Name : Martha, IP : 10.1.3.0
You can now send and receive messages.
Bob/Hello to Bob from Alice
Martha/Hello to Martha from Alice
Bob : Hello to Alice from Bob
Martha : Hello to Alice from Martha
Connection with Bob timed out
Connection with Martha timed out
Bob/Hello to Bob from Alice
Martha/Hello to Martha from Alice
Bob : Hello to Alice from Bob
Martha : Hello to Alice from Martha

root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Peer #3, Name : Martha, IP : 10.1.3.0
You can now send and receive messages.
Alice : Hello to Bob from Alice
Alice/Hello to Alice from Bob
Martha/Hello to Martha from Bob
Martha : Hello to Bob from Martha
Connection with Alice timed out
Connection with Martha timed out
Alice : Hello to Bob from Alice
Alice/Hello to Alice from Bob
Martha/Hello to Martha from Bob
Martha : Hello to Bob from Martha

root@abhishek: /home/abhishek/NetworksLab/Peer_to_Peer_chat
Peer #3, Name : Martha, IP : 10.1.3.0
You can now send and receive messages.
Alice : Hello to Martha from Alice
Bob : Hello to Martha from Bob
Alice/Hello to Alice from Martha
Bob/Hello to Bob from Martha
Connection with Alice timed out
Connection with Bob timed out
Alice : Hello to Martha from Alice
Bob : Hello to Martha from Bob
Alice/Hello to Alice from Martha
Bob/Hello to Bob from Martha
```

Timed out connections get re-established when one of the peers tries to send a message to the other.

Code Documentation:

1. We used an array of 'struct peer' which has the following form, to store data about the peers.

```
typedef struct {  
    char * name;  
    char * ip;  
    int port;  
    int socket_fd;  
    float latest_interaction_time;  
    struct sockaddr_in peeraddr;  
} peer;
```

2. A TCP Server is set up on each peer listening on port 6969 for new connections.
3. Each peer continuously calls the select system call and checks 3 file descriptors in the given order:
 - a. **server socket** : It checks if any new connection request has arrived on the socket. If it has, it processes it and accepts the connection if error free.
 - b. **stdin** : The process keeps checking stdin to see if the peer on which it is running wishes a message to some other peer. If there is a message, it separates out the peer name, and sends the message to the socket of the peer. If the connection does exist, a new connection is established, with a valid peer.
 - c. **client sockets**: the process keeps checking the client sockets to see if an active client has sent any new message or not. If a message is received, it prints it on stdout.
4. A separate thread is run to monitor the timeout of various peers. This was necessary since the main thread was getting blocked several times due to read and write operations. The timeout time is set to be 120 s.
5. Timed out connections have to re-establish connections when new messages need to be sent. Thus, a peer who tries to re-establish a connection may get a socket different from what it had previously.

Compilation and running procedure:

1. To compile use the **make** command in the same directory as the C++ file. A makefile has been provided.
2. To run the executable use **./P2P** command on each peer.
3. Edit the file **peer_details.txt** based on your peers and corresponding IP addresses. The format of peer_details.txt is given on Pg.1.
4. A copy of this process and file must be present on each peer, who needs to run it separately.