

Lab3 Socket Programming

孙济宸 520030910016

1. TCP chatroom

I used poll() to poll from stdin and TCP socket to allow client to both recv()ing incoming messages from socket and send()ing messages by taking keyboard input to the socket. The socket type is SOCK_STREAM for TCP. For this lab, the IP address of each client is hard-coded, and one TCP connection is only used by one message.

The image displays four terminal windows arranged in a 2x2 grid, each representing a different host in a TCP chatroom simulation. The windows are titled "host: h1", "host: h2", "host: h3", and "host: h4". Each window shows a sequence of messages sent and received between the hosts, including greetings, a poem, and a test message. The messages are formatted with "From" and "To" fields indicating the source and destination of the communication. The simulation demonstrates the use of poll() to handle multiple sockets and stdin simultaneously, allowing for bidirectional communication in a chatroom environment.

```
(base) root@madcreeper-VirtualBox:~/projects/computer-networks/lab/lab3# ./client
To h2 Hello!
To h3 114514
-----
From h2(10.0.0.2:56934)
> Hi!!!!
-----
From h4(10.0.0.4:47346)
> TEST123ABCD+~/*
To h4 This is a TCP chat room :)

(host: h1)

(host: h2)
From h1(10.0.0.1:45132)
> Hello!
-----
From h4(10.0.0.4:41098)
> $$$(!*$!*$$*!$(
To h1 Hi!!!!
To h3 This is a h2, hello h3!
-----
From h3(10.0.0.3:52116)
> This is h3, hi h2!

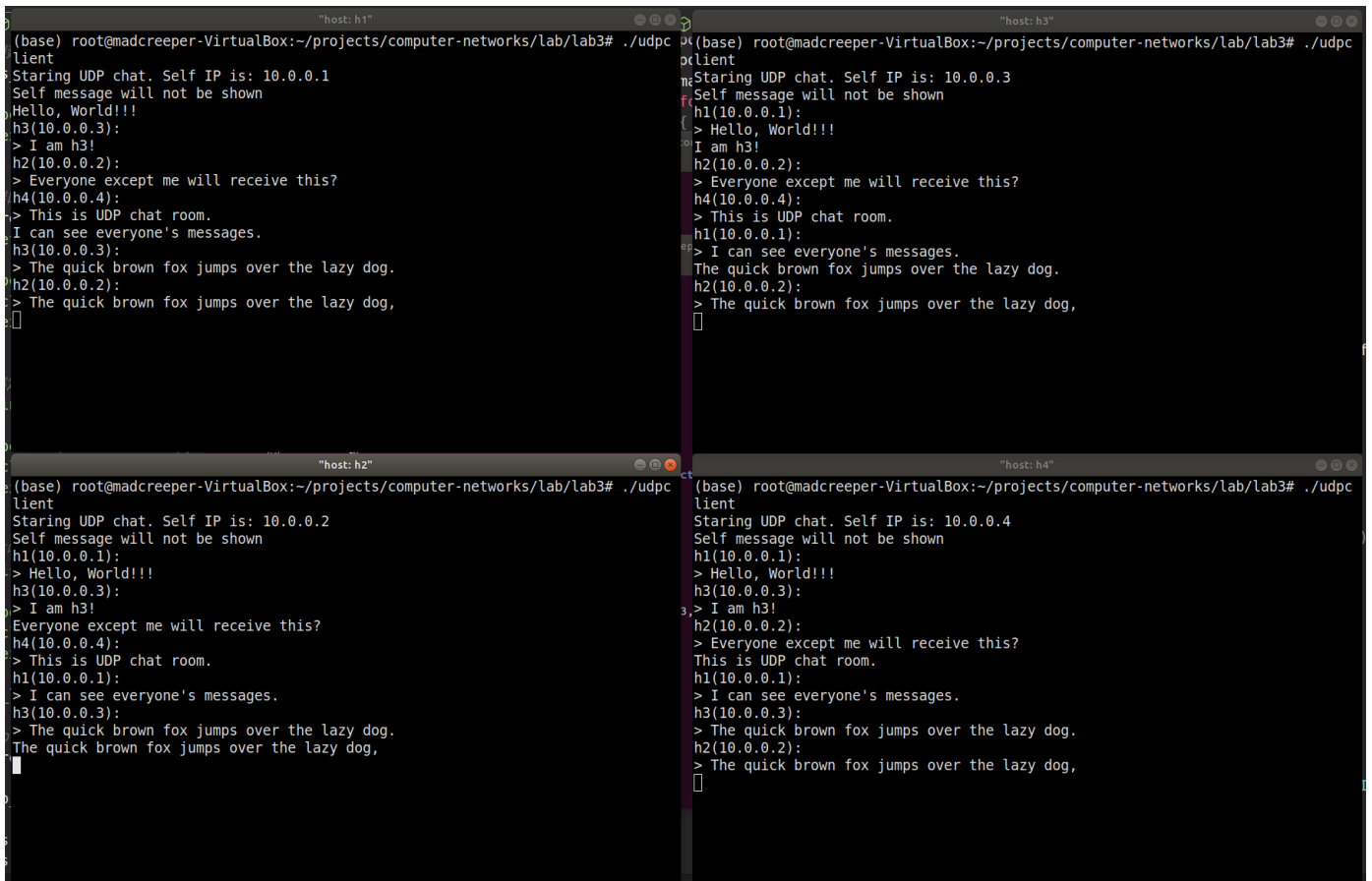
(host: h3)
From h1(10.0.0.1:45574)
> 114514
To h4 The quick brown fox jumps over the lazy dog.
-----
From h2(10.0.0.2:60364)
> This is a h2, hello h3!
To h2 This is h3, hi h2!

(host: h4)
From h3(10.0.0.3:34640)
> The quick brown fox jumps over the lazy dog.
To h2 $&$(!*$!*$$*!$(
To h1 TEST123ABCD+~/*
-----
From h1(10.0.0.1:34982)
> This is a TCP chat room :)

(host: h4)
```

2. UDP chatroom

For UDP chatroom, `poll()` is also used. The socket type is `SOCK_DGRAM` for UDP. Messages are sent via `sendto()` and `recvfrom()` sockets that are bound to the broadcast IP, in this case 10.255.255.255. For filtering out the client's own messages from showing, I used `getifaddrs()` so each client knows its own ip and can filter the messages by sender IP.



```
(base) root@madcreeper-VirtualBox:~/projects/computer-networks/lab/lab3# ./udpc
lient
Starting UDP chat. Self IP is: 10.0.0.1
Self message will not be shown
Hello, World!!!
h3(10.0.0.3):
> I am h3!
h2(10.0.0.2):
> Everyone except me will receive this?
h4(10.0.0.4):
> This is UDP chat room.
I can see everyone's messages.
h3(10.0.0.3):
> The quick brown fox jumps over the lazy dog.
h2(10.0.0.2):
> The quick brown fox jumps over the lazy dog,
[]

(host: h1)

(base) root@madcreeper-VirtualBox:~/projects/computer-networks/lab/lab3# ./udpc
lient
Starting UDP chat. Self IP is: 10.0.0.3
Self message will not be shown
h1(10.0.0.1):
> Hello, World!!!
I am h3!
h2(10.0.0.2):
> Everyone except me will receive this?
h4(10.0.0.4):
> This is UDP chat room.
h1(10.0.0.1):
> I can see everyone's messages.
The quick brown fox jumps over the lazy dog.
h2(10.0.0.2):
> The quick brown fox jumps over the lazy dog,
[]

(host: h3)

(base) root@madcreeper-VirtualBox:~/projects/computer-networks/lab/lab3# ./udpc
lient
Starting UDP chat. Self IP is: 10.0.0.2
Self message will not be shown
h1(10.0.0.1):
> Hello, World!!!
h3(10.0.0.3):
> I am h3!
Everyone except me will receive this?
h4(10.0.0.4):
> This is UDP chat room.
h1(10.0.0.1):
> I can see everyone's messages.
h3(10.0.0.3):
> The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog,
[]

(host: h2)

(base) root@madcreeper-VirtualBox:~/projects/computer-networks/lab/lab3# ./udpc
lient
Starting UDP chat. Self IP is: 10.0.0.4
Self message will not be shown
h1(10.0.0.1):
> Hello, World!!!
h3(10.0.0.3):
> I am h3!
h2(10.0.0.2):
> Everyone except me will receive this?
This is UDP chat room.
h1(10.0.0.1):
> I can see everyone's messages.
h3(10.0.0.3):
> The quick brown fox jumps over the lazy dog.
h2(10.0.0.2):
> The quick brown fox jumps over the lazy dog,
[]

(host: h4)
```