

«به نام پروردگار»

گزارش پروژه اول شبکه های کامپیوتری

«پیاده سازی پیام رسان ساده»

استاد: دکتر پویا حجازی

دانشجو: محسن محمدیان

شماره دانشجویی: 9831502

بررسی اجمالی:

در این پروژه یک پیام رسان ساده را پیاده سازی کرده ایم که انتقال پیام ها میان کاربرها از طریق سرور انجام می شود. کاربرها پیام خود را، که می تواند فایل یا متن باشد، به سرور ارسال می کنند و سرور آن پیام را دریافت کرده و به کاربر مقصد ارسال می کند.

برای پیاده سازی این روند از TCP socket programming استفاده شده است. به این صورت که سرور یک سوکت را ایجاد می کند و کلاینت ها با استفاده از IP address و port خود و سرور، با سرور ارتباط برقرار می کنند. چون در ارتباطات TCP ما می توانیم چندین سوکت در سمت سرور داشته باشیم، پس می آیم از multithread programming استفاده می کنیم و برای هر ارتباطی که هر کلاینت با سرور برقرار می کند یک thread برای مدیریت کردن آن ارتباط ایجاد می کنیم. در سمت کلاینت نیز یک سوکت ایجاد کرده و به سرور درخواست اتصال می دهیم.

توابع پیاده سازی شده:

ابتدا توابع موجود در سرور را یک بررسی مختصر می کنیم و سپس به بررسی توابع کد کلاینت می پردازیم.

main():

این تابع شروع کار ما هست و برنامه ی سرور از این نقطه آغاز می شود. در این تابع هست که ما آدرس IP و پورت سرور را ایجاد می کنیم و آنرا bind می کنیم تا در تابع start منتظر اتصالات کلاینت ها بمانیم. در این تابع همچنین ما یک thread از تابع get_data() ایجاد می کنیم و این thread تا هر زمان که thread اصلی (main thread) فعال باشد فعال خواهد بود. کار این thread آن هست که همیشه یک ورودی با دستور GET INFO می گیرد و اطلاعات موجود در جدول router را به ما بر می گرداند.

start():

در این تابع سرور منتظر درخواست های کلاینت ها برای ارتباط می ماند و هر زمان که یک کلاینت درخواست داد، برای آن ارتباط یک thread از handle_client ایجاد می کند.

handle_client():

این تابع از اصلی ترین توابع پیاده شده در کد سرور هست. چراکه مدیریت ارتباطات با کلاینت ها را انجام می دهد. در این تابع ابتدا یک query به DB زده می شود و اطلاعات کاربران دریافت می شود و سپس نام کلاینت تازه وارد دریافت می شود. هنگام دریافت نام، تا زمانی که نام تکراری دریافت کند، پیغام Duplicate username را برای کلاینت می فرستد و از او می خواهد نام دیگری را برگزیند. سپس اگر نام دریافتی تایید شد، یک پیغام Accept برای کلاینت می فرستد، نام و اطلاعات او را در DB ثبت می کند و به سایر کلاینت ها اطلاع می دهد که کاربری با این نام به سیستم پیوسته است.

همچنین هنگامی که یک کاربر به سیستم می پیوندد، لیست کاربران برای او ارسال می گردد تا بتواند انتخاب کند با چه کسی تمایل به chat کردن هست.

در این تابع یک حلقه "while connected" داریم، که این حلقه تا زمانی پابرجاست که از کلاینت پیغام "DISCONNECT" را دریافت نکند؛ پس از دریافت پیغام "DISCONNECT" سرور اطلاعات کاربر را از جدول مسیریابی حذف می کند و به سایر کاربران اطلاع می دهد که کاربر با این نام از سیستم خارج شد.

در این حلقه همچنین نام کاربر مقصد گرفته شده و تابع connect_to_dest_client فراخوانی می شود، که سرور به گونه ای به کلاینت گیرنده متصل می شود و این احساس را به کلاینت فرستنده می دهد که خود گویی به کلاینت گیرنده متصل شده است.

:send_file()

این تابع برای ارسال فایل هست که کد مشترکی هم در برنامه سرور و هم در برنامه کلاینت دارد. این تابع یک connection و یک file name را می گیرد و سپس برای کاربر فایل را ارسال می کند. توجه شود که سرور ابتدا فایل را از یک کاربر دریافت کرده و سپس برای کلاینت گیرنده ارسال می کند. حال سرور فایل هایی را که دریافت می کند در یک path name خاص نگه داری می کند. مثلاً در اینجا Desktop به عنوان default به آن داده شده است. سپس در آن path به دنبال نام فایل گشته و برای کلاینت مقصد فایل موردنظر را ارسال می کند.

:receive_file()

این تابع برای دریافت فایل هست که این تابع نیز کد مشترکی هم در برنامه سرور و هم در برنامه کلاینت دارد. این تابع یک connection و یک file name را می گیرد و فایل دریافتی از کلاینت فرستنده را در path خود ذخیره می کند.

:send_message()

این تابع برای مدیریت پیام های از نوع text میان کلاینت فرستنده و گیرنده هست. زمانی که یک کلاینت مشخص می کند قصد ارسال پیام به چه شخصی را دارد، سپس مشخص می کند که تمایل به ارسال فایل دارد یا متن، اگر "Chat" را انتخاب کند، این تابع فراخوانی می شود و تازمانی که دستور "Quit" را وارد نکرده می تواند به شخص موردنظرش پیام بفرستد.

توجه شود که پس از ارسال "Quit" کلاینت تنها از ارسال پیام متنی خارج می شود و هنوز به آن کلاینت متصل هست و می تواند دوباره دستور "Chat" را وارد کند و با او پیام رد و بدل کند یا دستور "Send file" را وارد کند و برای کلاینت گیرنده فایل ارسال کند یا دستور "Quit" را دوباره وارد کند تا بتواند نام کلاینت دیگری را وارد کرده و با شخص دیگری پیام رد و بدل کند.

:connect_to_dest_client()

این تابع یک احساس را به کلاینت فرستنده می دهد که به کلاینت گیرنده متصل شده است. (در واقع همان اتصال مجازی یا virtual) در این تابع نام کلاینت گیرنده در جدول مسیریابی جستجو می شود و اگر موجود نباشد، برای فرستنده پیغام "Person is not exist" ارسال می شود.

مهم ترین بخش این تابع حلقه ی "while connected" آن هست که پس از آنکه فرستنده به گیرنده متصل شد، می تواند برای او فایل یا پیام متنی ارسال کند. اگر فرستنده دستور "Chat" را وارد کند، تابع send_message() فراخوانی می شود و اگر دستور "Send file" را ارسال کند، سرور نام فایل را از او می گیرد و سپس تابع receive_file() فراخوانی می شود و سرور پس از آنکه فایل را دریافت کرد، یک پیام برای گیرنده می فرستد که فایلی برای دریافت داری با عنوان "Send file" و گیرنده نیز به او پاسخ دریافت را می دهد و سرور شروع به ارسال فایل می کند.

دستور "Quit" نیز ارتباط مجازی کلاینت فرستنده با گیرنده را قطع می کند و فرستنده به تابع handle_client() باز می گردد تا به شخص دیگری پیام بفرستد یا از سیستم خارج شود.

:send_list()

این تابع لیست نام کاربرانی که در سیستم هستند را برای یک کلاینت تازه وارد می فرستد. که این اطلاعات را از جدول مسیریابی بدست می آورد.

:get_data()

درباره این تابع توضیح دادیم که به صورت thread ایجاد می شود و هرگاه کاربر سرور قصد گرفتن اطلاعات جدول مسیریابی را داشته باشد، با وارد کردن دستور "GET INFO" می تواند آنها را بدست آورد.

:Leave_alert()

این تابع زمانی که شخصی با دستور "DISCONNECT" سیستم را ترک کند، به سایر کلاینت ها اطلاع می دهد که او ارتباطش با سیستم را قطع کرده است.

توابع برنامه کلاینت:

روند این توابع در میان توابع سرور توضیح داده شد. اما یک توضیح مختصر برای درک بهتر درباره آنها داده می شود.

main():

این تابع اصلی و نقطه شروع برنامه ی کلاینت هست که یک سوکت ایجاد کرده و با استفاده از آدرس IP و پورت به سرور درخواست connection می دهد.

در این تابع کلاینت ابتدا نام خود را وارد می کند که اگر تکراری و در سرور موجود باشد، ناچار به انتخاب نامی دیگر هست. سپس تابع `get_list()` فراخوانی می شود که لیستی از نام کلاینت های محود دیگر در سیستم را کلاینت می دهد. (نظیر تابع `send_list` در سمت سرور)

سپس یک `thread` از تابع `listen_for_message()` ایجاد می شود، که این `thread` تا زمانی که `main` اجرا می شود، پا برجا هست و همواره در حال شنود پیام های دریافتی از سمت سرور هست.

در انتهای این تابع یک حلقه ی `while message != "DISCONNECT"` داریم که نام کلاینت مقصد را از کاربر می گیرد و برای سرور ارسال می کند و تا زمانی که کاربر دستور `"DISCONNECT"` را وارد نکرده حلقه برقرار هست.

پس از آنکه نام کلاینت مقصد را گرفت و از طرف سرور متوجه شد کلاین موجود هست، تابع `chat()` فراخوانی می شود. که در این تابع کلاینت می تواند با سه دستور `"Send file"` فایل ارسال کند، با دستور `"Chat"` شروع به ارسال پیام های متنی کند یا با دستور `"Quit"` از ارتباط با کلاینت مقصد خارج شده و دوباره به حلقه ی

`"while message != "DISCONNECT"` تابع `main` بازگردد و شخص دیگری را برای رد و بدل کردن پیام انتخاب کند.

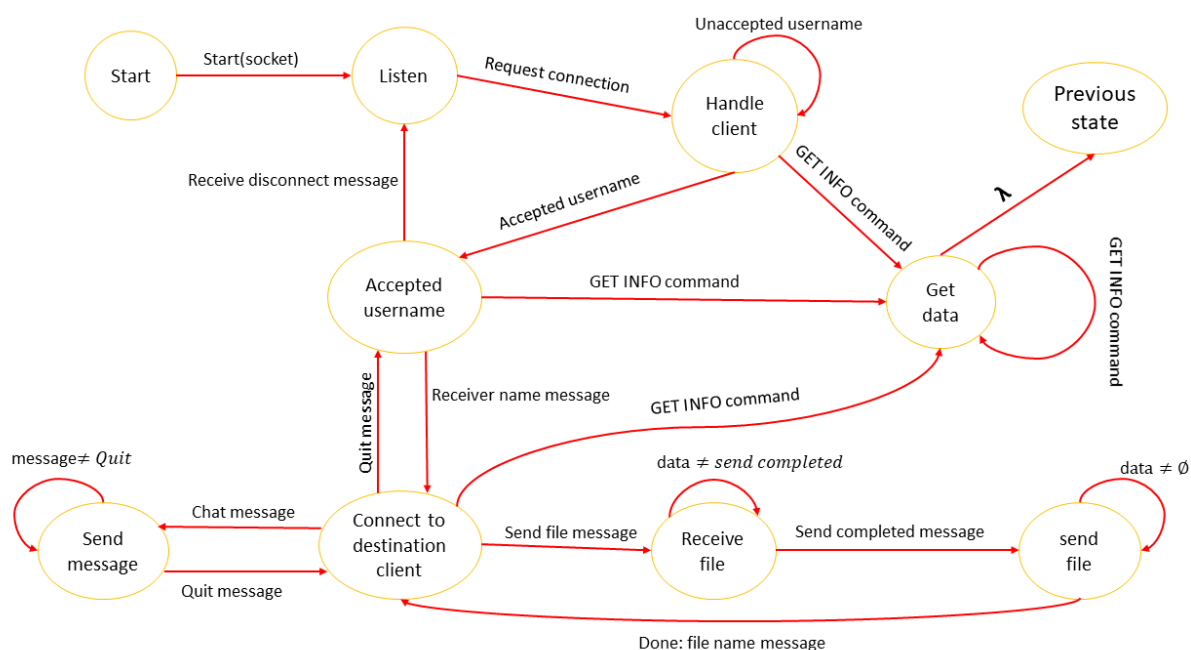
receive_file(), send_file():

دو تابع `send_file()` و `receive_file()` نیز روندشان در کد سرور توضیح داده شد. تنها تفاوتی که هست؛ آن هست که در تابع `send_file()` کلاینت، یک پارامتر دیگر بنام `path` داریم که `directory` فایل موردنظر برای ارسال را مشخص می کند.

str_msg():

این تابع برای ارسال پیام های متنی کلاینت هست و در حلقه تابع `chat()` زمانی که کاربر دستور `"Chat"` را وارد کند، فراخوانی می شود. (گونه ای نظری تابع `send_message()` در سرور هست)

FSM مربوط به سرور:



در این دیاگرام، منظور از Previous state، وضعیت قبلی پیش از وارد کردن دستور "GET INFO" می باشد. (به دلیل آنکه در تصویر جا نمی شد تمامی فلش های از یک وضعیت به وضعیت Get data را رسم کنیم، از این state استفاده کردیم)

نمونه گزارش تصویری:

```

Run: C:\Users\AJAMG\AppData\Local\Programs\Python\Python38\python.exe D:/Uni/terms/network/Projects/Pr2/c2.py
[Client connects] Client is connecting...
Enter your name
mohsen
You can contact to:
Enter name of person you want to send message to it.
niusha is joined to system now!
ali is joined to system now!
niusha
you can send message to niusha now.
You can send text message to someone with command "Chat" or send file with "Send file" command
What you want to do?
chat
Type something for sending...
hi!!

[2020-12-07 01:30:42] mohsen: hi!!
[2020-12-07 01:31:00] niusha: heyy
[2020-12-07 01:31:09] niusha: how r u?
I'M fine thanks
[2020-12-07 01:31:20] mohsen: I'M fine thanks
bye
  
```

```
Run: C:\Users\VAJAMG\AppData\Local\Programs\Python\Python38\python.exe D:/Uni/term3/network/Projects/Pr2/c1.py
[Client connects] Client is connecting...
Enter your name
>
You can contact to:
mohsen
Enter name of person you want to send message to it.
>
ali is joined to system now!

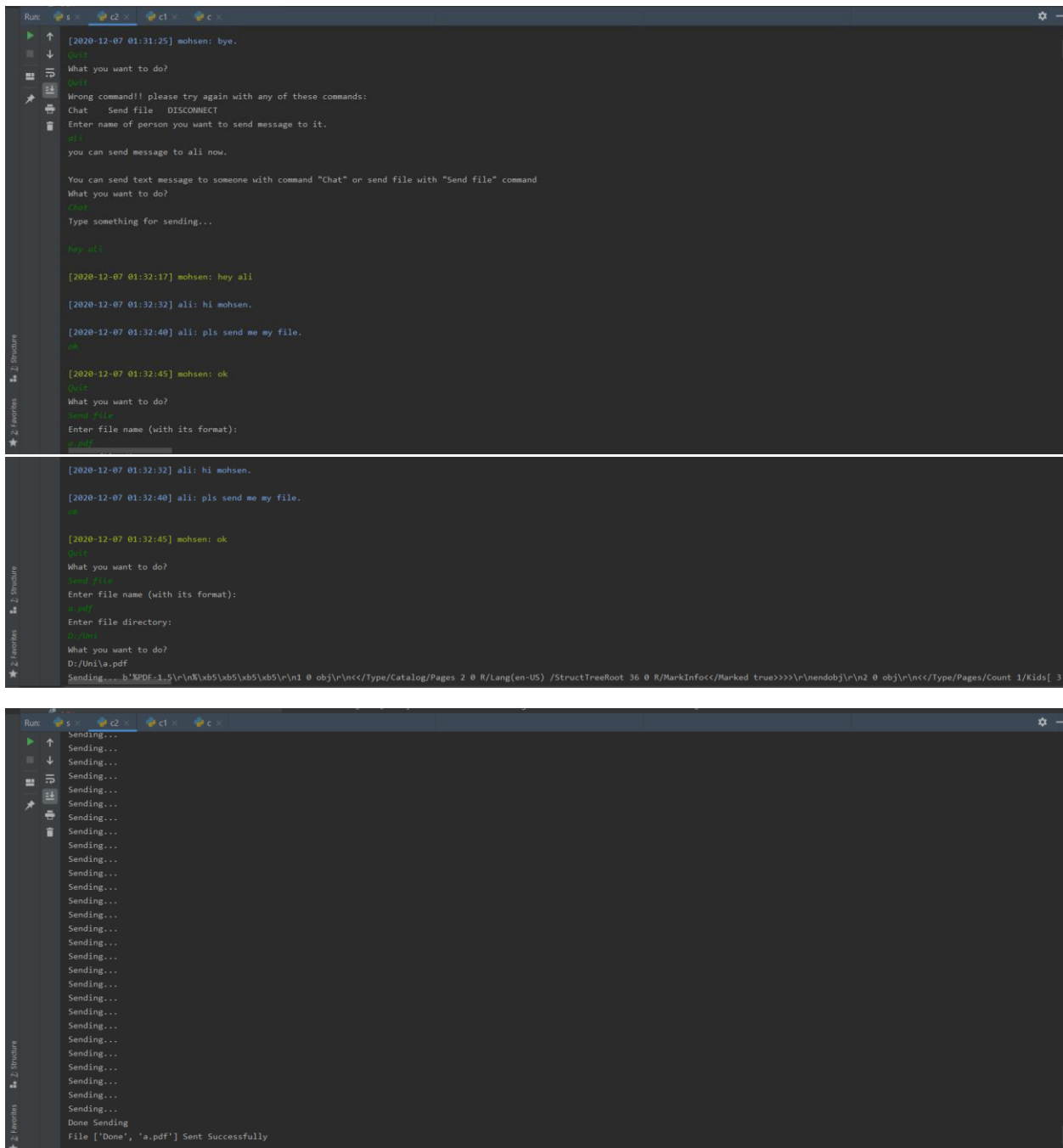
[2020-12-07 01:30:42] mohsen: hiii
>
Person is not exist
Your desired person is not exist!....try another person
Enter name of person you want to send message to it.
>
mohsen
you can send message to mohsen now.

You can send text message to someone with command "Chat" or send file with "Send file" command
What you want to do?
>
Type something for sending...
>

[2020-12-07 01:31:00] niusha: heyy
>
[2020-12-07 01:31:09] niusha: how r u?
```

```
Run: C:\Users\VAJAMG\AppData\Local\Programs\Python\Python38\python.exe D:/Uni/term3/network/Projects/Pr2/c.py
[Client connects] Client is connecting...
Enter your name
>
You can contact to:
mohsen niusha
Enter name of person you want to send message to it.
>
[2020-12-07 01:32:17] mohsen: hey ali
>
you can send message to mohsen now.

You can send text message to someone with command "Chat" or send file with "Send file" command
What you want to do?
>
Type something for sending...
>
[2020-12-07 01:32:32] ali: hi mohsen.
>
[2020-12-07 01:32:40] ali: pls send me my file.
>
[2020-12-07 01:32:45] mohsen: ok
1024
Receiving...
Done Receiving
What you want to do?
>
```




```

Run: s x c2 x c1 x c x
Type something for sending...

[2020-12-07 01:32:32] ali: hi mohsen.
[2020-12-07 01:32:32] mohsen: hi ali

[2020-12-07 01:32:40] ali: pls send me my file.

[2020-12-07 01:32:45] mohsen: ok
1024
Receiving...
Done Receiving
Quit
What you want to do?
Quit
Wrong command!! please try again with any of these commands:
Chat Send file DISCONNECT
What you want to do?

Person is not exist
Quit
Wrong command!! please try again with any of these commands:
Chat Send file DISCONNECT
Enter name of person you want to send message to it.

Person is not exist
[2020-12-07 01:32:45] mohsen: ok
Process finished with exit code 0

```

```

Run: s x c2 x c1 x c x
Sending...
Sending...
Sending...
Sending...
Sending...
Sending...
Done Sending
File ["Done", "a.pdf"] Sent Successfully

[2020-12-07 01:33:39] niusha: are u there?
Quit
Wrong command!! please try again with any of these commands:
Chat Send file DISCONNECT
Enter name of person you want to send message to it.
niusha
you can send message to niusha now.

You can send text message to someone with command "Chat" or send file with "Send file" command
What you want to do?
Chat
Type something for sending...
yes

[2020-12-07 01:34:10] mohsen: yes
I am

[2020-12-07 01:34:13] mohsen: i am.
ali left the system right now!

```

Server side:

```

Run: c2 x c1 x c x
handle_client RECEIVE FILE
52 5
54 a.pdf
C:\Users\ANAJANG\Desktop\1.pdf
Sending... b"%PDF-1.5\r\n\xB5\xB5\xB5\r\n1 0 obj\r\n<</Type/Catalog/Pages 2 0 R/Lang(en-US) /StructTreeRoot 36 0 R/MarkInfo<</Marked true>>>\r\nendobj\r\n2 0 obj\r\n<</Type/Pages/Count 1/Kids[
Done Sending
83 10
85 ["Done", "a.pdf"]
File a.pdf Smt Successfully
[MESSAGE RECEIVED] are u there?
handle_client niusha
[["all", "192.168.56.1", "10562"), ("mohsen", "192.168.56.1", "10541"), ("niusha", "192.168.56.1", "10547")]
[MESSAGE RECEIVED] yes
[MESSAGE RECEIVED] i am.
[MESSAGE RECEIVED] Quit
192.168.56.1
{"all", "192.168.56.1", "10562")
{"mohsen", "192.168.56.1", "10541")
{"niusha", "192.168.56.1", "10547")
handle_client qUIT
[["all", "192.168.56.1", "10562"), ("mohsen", "192.168.56.1", "10541"), ("niusha", "192.168.56.1", "10547")]
Person is not exist in data base!!
handle_client Quit
[["all", "192.168.56.1", "10562"), ("mohsen", "192.168.56.1", "10541"), ("niusha", "192.168.56.1", "10547")]
Person is not exist in data base!!
handle_client DISCONNECT
1 record(s) deleted
192.168.56.1
{"mohsen", "192.168.56.1", "10541")
{"niusha", "192.168.56.1", "10547")
192.168.56.1
192.168.56.1
192.168.56.1
Done Receiving

```