گزارش کار:در این پروژه با استفاده از پروتکل TCP به پیاده سازی SOCKET پرداختیم.این پروژه شامل دو فایل client.py و server.py است که در مورد روند پیاده سازی هر کدام توضیح می دهیم:

- Client.py :در این فایل ابتدا از کاربر دو ورودی name, path را دریافت می کنیم و سپس این دو مقدار را به تابع socket می دهیم این تابع به گونه ای پیاده سازی شده است که ابتدا یک شی از کلاس socket ایجاد می کند سپس نام میزبان را دریافت می کند. در مرحله ای بعد پورت دلخواهی را تنظیم می کنیم.با استفاده از متد connect از کلاس socket شروع به اتصال می کنیم(با استفاده از نام میزبان و پورت انتخاب شده)و سپس با استفاده از متد Send شروع به ارسال پیام می کنیم در این قسمت باید پیام به صورت باینری در بستر شبکه ارسال شود پس از متد encode استفاده می کنیم.سپس با استفاده از متد recv پیام تایید برقراری ارتباط یعنی (200 ok) را دریافت کرده و سپس شروع به خواندن محتوایات ارسالی توسط سرور می کنیم بازهم با استفاده از متد recv شروع به دریافت پیام می کنیم در این مرحله باید به این نکته توجه کرد که محتویات پیام ما به صورت باینری است به همان دلیل که گفته شد پس با استفاده از متد decode فایل دریافتی را به حالت ابتدایی برگردانده و در مسیر دلخواه کاربر که در ابتدا از آن دریافت کرده بودیم فایلی ایجاد کرده و در آن ذخیره می کنیم.

- Server.py:خیلی شبیه به client دقیقا یک شی از کلاس socketایجاد می کنیم. سپس نام میزبان را دریافت می کنیم. در مرحله ای بعد پورت دلخواهی را تنظیم می کنیم.با استفاده از متد bind کلاینت رو متصل می کنیم به Server سپس با متد listen منتظر در خواست کاربر می شویم و سپس متد accept یک شی از

کلاینت می سازیم و با استفاده از آن شروع به دریافت پیام کرده و سپس به دنبال فایل مورد نظر می گردیم با دستورسیستمی(os.path.exists) و سپس با استفاده از متد Send شروع به ارسال پیام می کنیم در این قسمت باید پیام به صورت باینری در بستر شبکه ارسال شود پس از متد encode استفاده می کنیم.

تمرین ۲

الف) طبق مطالب گفته شده در کلاس و همچنین ویژگی پروتکل TCP (connection oriented)می دانیم ابتدا اصطلاحا احوال پرسی می کند(hand shaking) و بعد کانکت می شود بنابراین با پیغام خطای ConnectionRefusedError مواجه می شویم.که دلیل این موضوع خودداری مقصد یا سرور از برقراری ارتباط است در واقع به دلیل توقف Server است.

ب)با توجه به ویژگی های پروتکل UDP (connection less) می دانیم که در این پروتکل اصلاحا احوال پرسی hand shaking نداریم و پیام client تحت هر شرایطی فرستاده می شود اگر نیاز به دریافتی از سرور نباد که هیچ اتفاقی نمیفته اما اگر نیاز به پاسخ از طرف Server باشد و به خاطر توقف server پیامی دریافت نمی کند اصطلاحا با خطای ConnectionResetError رو به رو می شویم.تفاوتی که بین دو سناریو مطرح شده در قسمت الف و ب وجود دارد در نحوه ارتباط آنهاست که اولی connection oriented و دومی connection less است.

ج)طبق ویژگی های UDP مشکل نمی خوریم زیرا در این پروتکل به محض اینکه پیام میفرستیم ارتباط برقرار می شود در واقع هر بار که پیام دادن شروع به اصطلاحا احوال پرسی می کنند.اما در TCP در همان ابتدا شروع به احوال پرسی می کنند و ارتباط برقرار می شود پس به خطای ConnectionRefusedError برخورد می کنیم و کلاینت متوقف می شود.

د)می دانیم socket tcp چهار حالت دارد ip مبدا و مقصد و پورت مبدا زیرا برای مالتی پلکس و دی ماتی پلکس کردن داده ها و تشخیص آنها از هم به این چهار آدرس نیاز داریم.اما در UDP فقط ip و پورت مقصد رو احتیاج داریم.نقش پورت مقصد در پروتکل TCP دی مالتی پلکس کردن هست نقش پورت مبدا پاسخ فرستادن وتاییدیه فرستادن است.اگر در UDP نیاز به پاسخ نباشد پورت مبدا بلااستفاده است.اگر نیاز به پاسخ باشد از پورت مبدا استفاده می کنیماز پورت مقصد هم برای دی مالتی پلکس کردن استفاده می کنیم.

ه)یکی دیگر از ویژگی های TCP گارنتی در ارسال داده هاست و اگر بافر پر شود داده ها با تاخیر و پس از خالی شدن بافر فرستاده می شود اما به دلیل نبود همچین ضمانتی در UTP بخشی از داده ها ممکن است از بین برود.

تمرین ۳

الف)به مشکل بر می خوریم و همزمان نمی توانیم روی یک پورت و یک آدرس IP بین دو برنامه connection برقرار شود و این به خاطر ویژگی های TCP است.زیرا اصطلاحا یک server از روی یک پورت داره listen می کند و server دیگر به طور همزمان نمی تواند این کار را انجام بدهد.

ب)به مشکلی بر نمی خوریم و برنامه ها با موفقیت اجرا می شوند و درواقع پهنای پورت برای هر پروتکل متفاوت کاملا مستقل است.