گزارش تمرین اول – مهندسی اینترنت

رامتین احسانی 97521018

فایل های حاوی کد این برنامه نوشته شده: Client1.py – Client2.py

کد های نوشته شده برای این دو فایل کاملا شبیه هم هستند با این تفاوت که برای راحت کردن فرآیند اجرای کد ها، دو کلاینت از هم جدا شده اند تا شماره port های منحصر به فردی به هرکدام تعلق بگیرد.

منطق برنامه به این صورت است که هر دو کلاینت در ابتدا بر روی دو سوکت شروع به کار میکنند. یک سوکت اختصاصی برای ارسال پیام به طرف مقابل در thread اصلی. یک سوکت برای گوش دادن به پیام های ارسالی در thread و پورت جداگانه. دو لیست در اختیار هر کاربر است: لیست پیام های ارسالی و لیست پیام های دریافتی. هر بار که پیامی برای طرف مقابل ارسال میکنند، لیست پیام های ارسالی آن ها افزایش میابد و هر بار پیامی دریافت میکنند لیست پیام های دریافتی افزایش میابد.

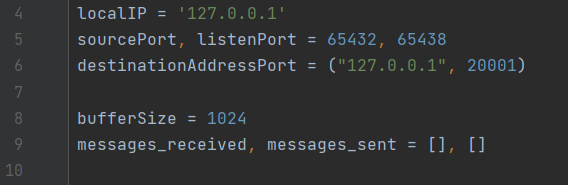
با ارسال پیام، علاوه بر پیام اصلی، عددی هم همراه با آن ارسال میشود که نشان دهنده تعداد پیام هایی است که با **موفقیت** از طرف مقابل دریافت کرده است. طرف مقابل(کاربر B) با دریافت پیام از کاربر A، عدد همراه با پیام را چک میکند. در صورتی که **تعداد** پیام های ارسالی که طرف مقابل(کاربر A) دریافت کرده است با تعداد پیام های نوشته شده توسط کاربر B برابر نباشد، کاربر B تمام پیام های نوشته شده را در سوکت و thread جدید برای کاربر A میفرستد.

پیام های دوباره ارسال شده با تگ “resend” فرستاده میشوند تا کاربر مقابل متوجه بشود.

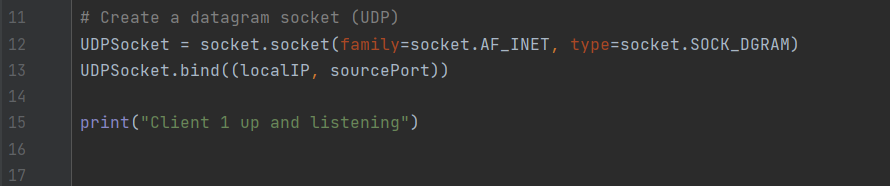
برنامه با استفاده از زبان pyhton و با استفاده از پکیج های threading و socket نوشته شده است.



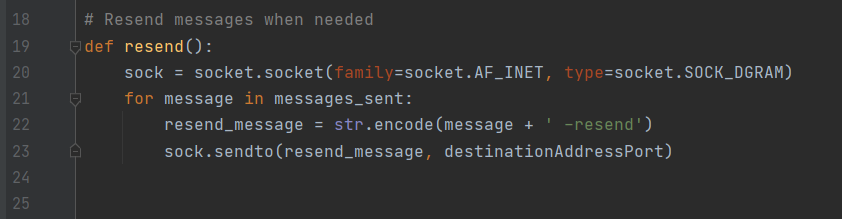
در قسمت اول کد، اطلاعات کلاینت مورد نظر مانند شماره پورت های لازم در ادامه برنامه، ip، bufferSize و لیست هایی برای ذخیره پیام های ارسالی و دریافتی تعریف شده اند. همچنین اطلاعات طرف مقابل که شامل ip و پورت **شنونده** طرف مقابل است هم در destinationAddressPort نوشته است.



در قسمت بعدی، سوکت برای کلاینت تعریف کردم که از این سوکت برای فرستادن پیام به طرف مقابل استفاده میشود. سوکت تعریف شده UPD است. (SOCK\_DGRAM)

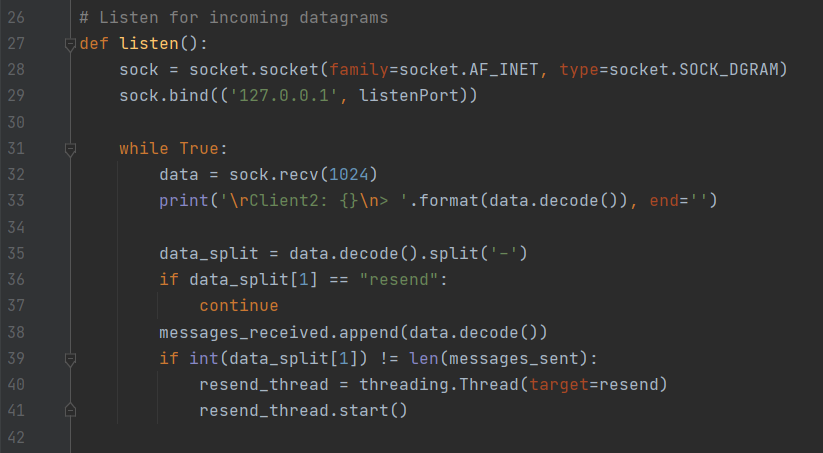


برای قسمت بعدی، تابع resend را نوشتم که در آن سوکت جدیدی تعریف شده است و در صورت فراخوانی **تمام** پیام های **نوشته** شده به طرف مقابل را دوباره ارسال میکند.

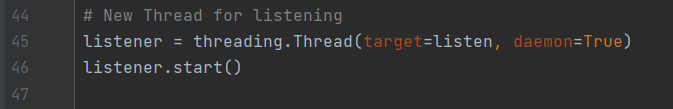


بعد از آن، تابع listen را نوشتم که بر روی listenPort همیشه در حال گوش دادن است. در همین تابع است که با دریافت پیام، عدد ارسالی هم چک میشود تا درصورت مغایرت، تابع resend را در thread جداگانه ای فراخوانی کند. پیام های دریافتی در صورتی که resend نباشند در لیست پیام های دریافتی ذخیره میشوند و نمایش هم داده میشوند.

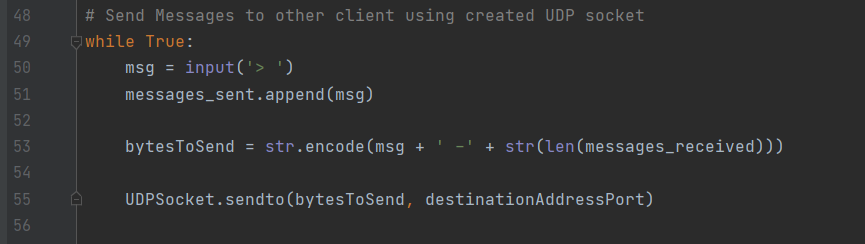
در صورتی که تگ “resend” داشته باشد کاری با آن پیام ندارد و فقط آن را نمایش میدهد.



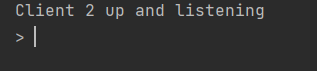
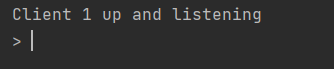
کار گوش دادن هم همانگونه که اشاره شد در thread جدایی انجام میگیرد.



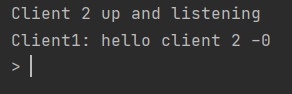
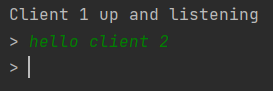
در thread اصلی هم با استفاده از سوکتی در اول تعریف کردم به پورتی که طرف مقابل در حال **گوش دادن** است پیام میفرستیم. پیام ها هم علاوه بر متن اصلی، حاوی تعداد پیام های دریافتی که موفق به کاربر رسیده اند نیز می باشد.



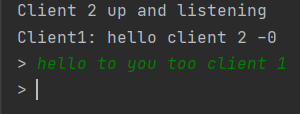
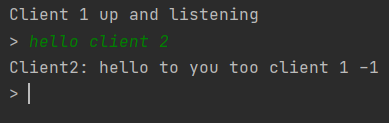
اجرای برنامه: بعد از اجرا هر دو کاربر آماده ارسال و دریافت پیام هستند.

ارسال پیام:

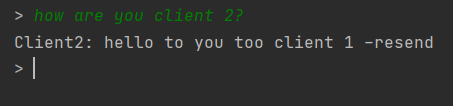
 

کلاینت 1 به 2 پیام ارسال کرده و برای کلاینت 2 به نمایش درآمده است. 0 یعنی هنوز از تو پیامی نگرفتم.

هر دو یک پیام ارسال کرده و یک پیام دریافت کرده اند.

برای تست قابلیت resend شرط چک کردن تعداد پیام ها را تغییر میدهم تا قطعا دوباره پیام ها را ارسال کند تا مطمئن بشویم که برنامه درست کار میکند:



با ارسال این پیام از 1 به 2، کلاینت 2 با توجه به عددی که دریافت میکند متوجه میشود که کلاینت 1 پیام قبلی را دریافت نکرده است. پس دوباره پیام ها را برای 1 میفرستد( “resend”).

پایان گزارش