به نام خدا

امیرحسین محمد امری

9527113

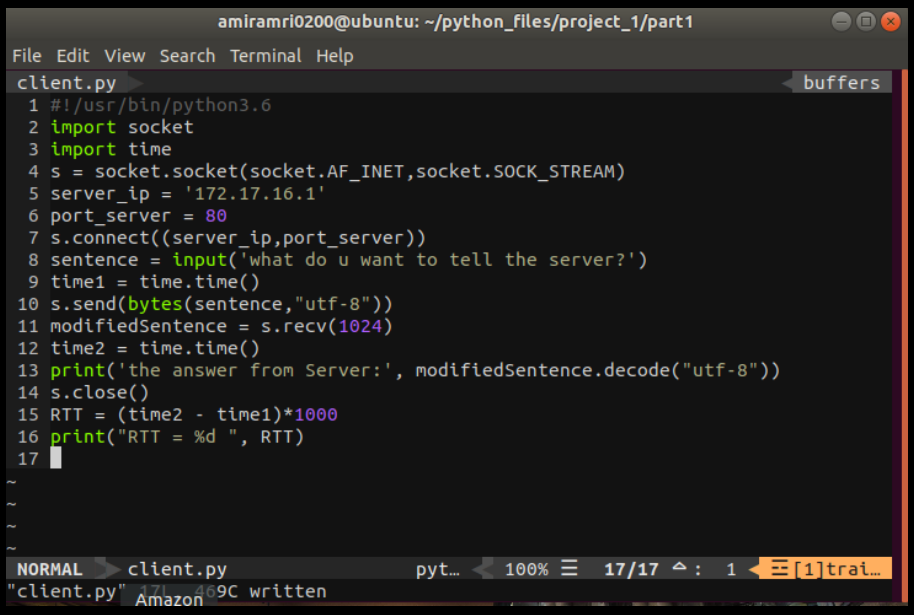
گزارش کار

تذکر : در قسمت توضیح کد ها صرفا کد هایی که در قسمت های قبل تکرار نشده و توضیح داده نشده را توضیح دادم و از تکرار بی مورد اجتناب کرده ام.

سوال ۱)

در این سوال قصد داریم یک پیام به سرور فرستاده و جواب ACK آن را دریافت و همچنین اختلاف زمانی بین درخواست و جواب را مشاهده کنیم (RTT) برای اینکار هم میتوانستیم از ماژول تایم در کد خود استفاده کنیم و هم از نرم افزار wireshark که من از روش اول مقدار RTT را بدست اوردم.

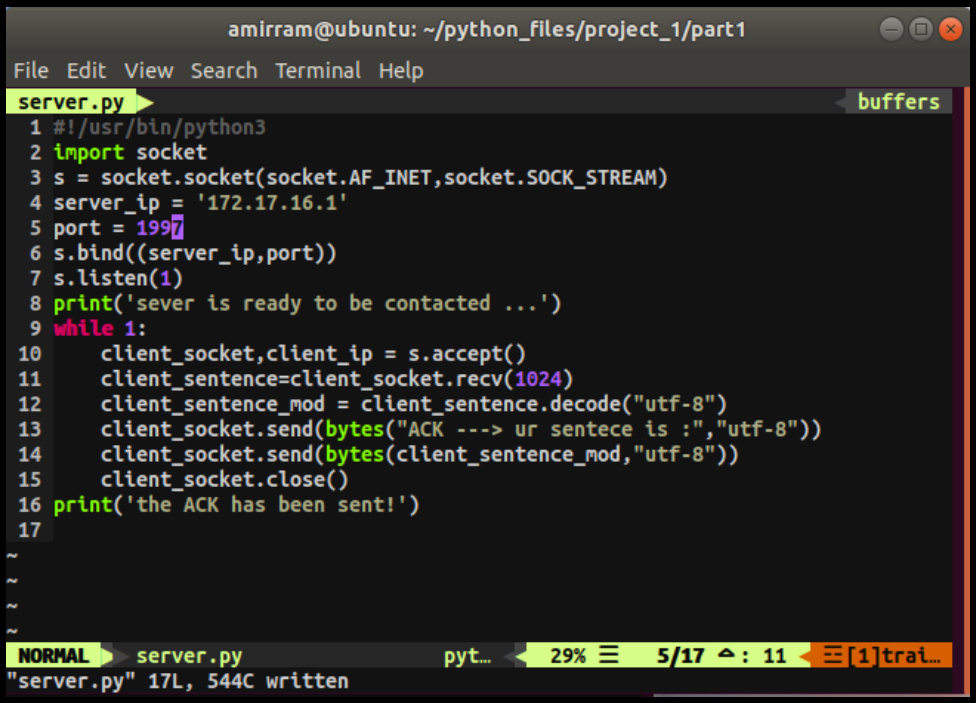
در تصویر زیر کد سمت کلاینت را میبینیم :



- در خط دوم و سوم کتابخانه های مورد نیاز را اضافه کردم .
- در خط چهارم یک شی (object) از نوع سوکت تعریف کردم
- در خط پنجم آدرس سرور را تعریف کردم و در خط ششم پورت سرور
- در خط هفتم سوکتس را ک تعریف کردم را به سرور وصل کردم تابع connect یک ورودی از جنس تاپل میگیرد که تاپل ما متشکل از ادرس و پورت سرور است .
- در خط ۸ یک جمله از ورودی دریافت میکنیم
- در خط ۹ توسط ماژول تایم زمان همان موقع سیستم را ذخیره میکنیم.
- در خط ۱۰ توسط تابع send جمله ای را که میخواهیم به سرور بفرستیم را ارسال میکنیم .توجه شود که این جمله باید از نوع utf-8 کد شود که این کار درون تابع send انجام شده
- در خط ۱۱ توسط تابع recv پکت های آمده از سمت سرور را دریافت میکنیم . توجه شود که عدد درون تابع recv حجم پکت های دریافتی را مشخص میکند.
- در خط ۱۲ زمان سیستم را بلافاصله بعد از دریافت جواب سرور ذخیره کردیم .

- در خط ۱۴ اتصال را بستیم
- در خط ۱۵ اختلاف زمان های سیستم اختلاف زمانی بین ارسال و دریافت جواب را نشان میدهد و برای این که این عدد به میلی ثانیه نمایش داده شود در ۱۰۰۰ ضرب شده

در تصویر زیر کد سمت کلاینت را مشاهده میکنید:



- توضیح در مورد ورودی های socket : AF_INET نمایانگر پروتکل ip4 و SOCK_STREAM نمایانگر اتصال TCP میباشد.
- خطوط ۱ تا ۵ مشابه قبل میباشد .
- در خط ۶ با دستور bind() سرور و پورت مربوط به آن را به وابسته میکنیم تا کلاینت بتواند به آن متصل شود.
- در خط ۷ به سرور میگوییم که منتظر دریافت پاسخ باشد .توجه شود که عدد درون آن مشخص میکند که همزمان با چند کلاینت در حال مکالمه باشد . در اینجا به عدد یک بسنده کردیم.
- در خط ۹ یک حلقه همواره درست را ایجاد کرده ایم به این منظور که سرور باید همواره منتظر دریافت درخواست باشد .
- در خط توسط تابع accept() درخواست اتصال را میپذیریم .این تابع دو خروجی دارد که یکی از آن ها یک شی از سوکت مشتری و دیگری ادرس مشتری میباشد که آن ها را همانطور که مشاهده میشود به ترتیب در متغییر های client_sock و client_ip ذخیره کردم.

خروجی کد را در تصویر زیر مشاهده میکنید :

```
amiramri0200@ubuntu: ~/python_files/project_1/part1
File  Edit  View  Search  Terminal  Help
amiramri0200@ubuntu:~/python_files/project_1/part1$ python3 client.py
what do u want to tell the server? hello world
the answer from Server: ACK ---> ur sentece is : hello world
RTT = 20 ms
amiramri0200@ubuntu:~/python_files/project_1/part1$
```

سوال ۲)

در این قسمت خواسته شده که یک وب سرور ایجاد شود که توانایی پردازش یک درخواست را داشته باشد .

کد سمت کلایت را در تصویر زیر مشاهده میکنید :

```
File  Edit  View  Search  Terminal  Help
http_client.py                                                    buffers
 1 from socket import *
 2 import webbrowser
 3 import sys
 4 server_ip = '172.17.16.1'
 5 file_path = "helloworld.html"
 6 target_port = 1997 # create a socket object
 7 try:
 8     client = socket(AF_INET, SOCK_STREAM)
 9
10     # connect the client
11     client.connect((server_ip, target_port))
12 except:
13     print("can't connect to the server")
14     sys.exit()
15
16 # send some data
17 data = ""
18 request = ("GET %s HTTP/1.1\r\nHost:%s\r\n\r\n" % (file_path,server_ip)).encode("utf-8")
19 client.send(request)
20 with open("helloworld.html", 'wb') as f:
21     while True:
22         data =(client.recv(1024)).decode("utf-8")
23         if data == "end connection\r\n":
24             f.close()
25             break
26         f.write(data.encode("utf-8"))
27 deliver_state =(client.recv(1024)).decode("utf-8")
28 print( "deliver_state = " + deliver_state[0: deliver_state.find("\r")])
29 client.close()
30 webbrowser.open('helloworld.html')
31 print("data connection disconnected")
32
~
~
~
NORMAL  http_client.py                    python  utf-8[unix]  31%  10/32 : 24
```

- در خط ۵ نام فایلی که قرار است از سرور بخواهیم ذخیره شده
- در خطوط ۷ تا ۱۴ به سرور وصل شدیم .فقط توجه شود که در قالب try: except: اگه کد های درون بدنه try نتوانند اجرا شود کد های دورن بدنه except اجرا میشوند.در اینجا اگه کلاینت به هر دلیلی نتواند به سرور وصل شود جمله ی cant connect to server چاپ میشود.
- در خط ۱۸ یک رشته به فرمت http ایجاد کردیم که در آن اسم فایلی که میخواهیم دریافت کنیم و آدرس سرور مورد نظر وجود دارد
- در خط ۲۰ از فرمت with open() as f : استفاده کردیم در حالت یک فایل با نامی که درون تابع open مینوسیم ایجاد و آن را باز میکند و وقتی از بدنه این فرمت خارج میشویم به صورت اتوماتیک فایل ایجاد شده را میبندد.توجه شود که درون بدنه اگر بخواهیم از کلاس های مربوط به فایل استفاده کنیم باید از f به جای اسم فایل استفاده کنیم .
- در خطوط بین ۲۱ تا ۲۶ درون یک حلقه ی همواره درست به طور مدام پکت هایی که از سمت سرور می آید را درون متغییر data ذخیره میکنیم و درصورتی که از سمت سرور پیامی حاوی اتمام اتصال دریافت کنیم فایل را میبندیم و از حلقه ی while خارج میشویم .
- و در خط ۳۰ صفحه ی وبی را که دریافت کرده ایم را باز میکنیم .

در تصویر زیر کد سمت سرور را مشاهده میکنید :

http_server.py

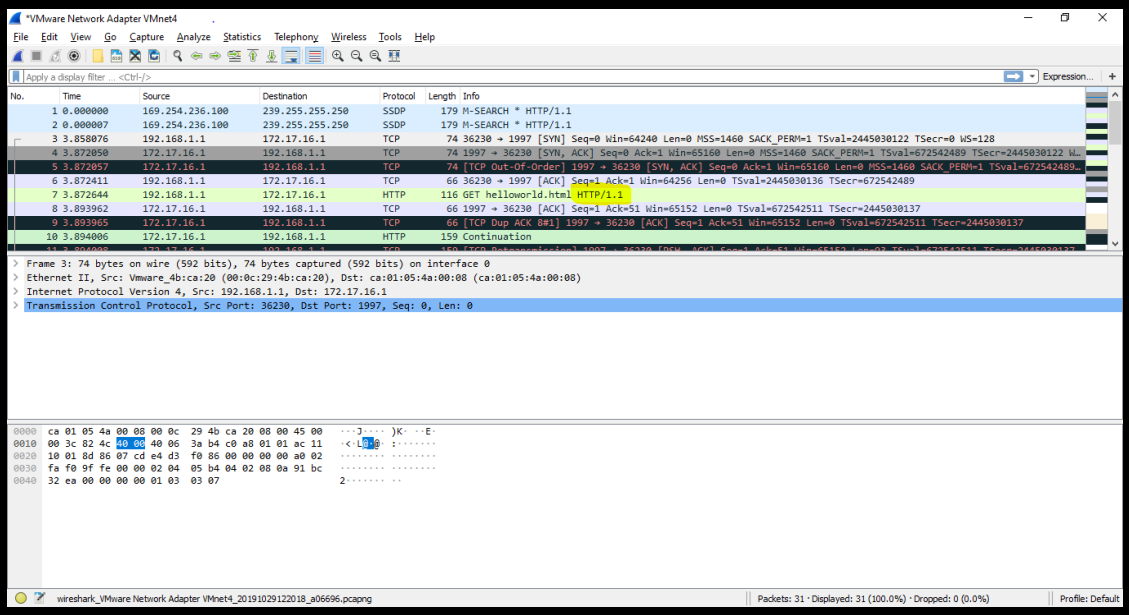
```
 4 import time
 5 serverPort = 1997
 6 server_Socket = socket(AF_INET,SOCK_STREAM)
 7 server_Socket.bind(('172.17.16.1', serverPort))
 8
 9 server_Socket.listen(1)
10 print ("server is ready to be contacted . . .\n")
11 while True:
12     client_Socket, addr = server_Socket.accept()
13     recieved_header = (client_Socket.recv(1024)).decode("utf-8")
14     end = recieved_header.rfind("HTTP")
15     first = recieved_header.find("GET")
16     temp  =recieved_header[first + 4:end-1]
17     file_name =temp[temp.rfind("/") + 1:]
18     error_file = "notFound.html"
19     try:
20         f = open(file_name, 'rb')
21         deliver_state = True
22     except:
23         f = open(error_file, 'rb')
24         deliver_state = False
25     while True:
26         temp1 = f.read(1024)
27         if not temp1:
28             time.sleep(0.1)
29             f.close()
30             client_Socket.send(("end connection\r\n").encode("utf-8"))
31             break
32         client_Socket.send(temp1)
33     if deliver_state:
34         deliver_msg = "HTTP/1.1 200 OK\r\n\r\n"
35     else:
36         deliver_msg = "HTTP/1.1 404 Not Found\r\n\r\n"
37     client_Socket.send((deliver_msg).encode("utf-8"))
38     print (deliver_msg[0: deliver_msg.find("\r")])
39     client_Socket.close()
40
NORMAL  http_server.py
```

- تا خط ۱۳ مشابه کد های قبلی است .اما در خط ۱۴ index مربوط به اخرین حرف قبل از کلمه ی HTTP را توسط تابع find() پیدا میکنیم همین کار را برای واژه ی GET انجام میدهیم .اگر به رشته ای که از سمت کلاینت فرستاده میشود نگاه کنیم متوجه میشویم که نام فایل دقیقا بین دو واژه گفته شده قرار دارد برای همین index های این دو واژه را بدست آوردیم .
- در خط ۱۶ به ایندکس GET عدد چهار را اضافه و از عدد ایندکس HTTP عدد یک را کم کردیم تا دقیقا به نام فایل بدون وجود هیچ فاصله ای برسیم.اکنون نام فایل در اختیار ماست.توجه شود که اگر همراه فایل مسیر مربوط به آن را هم بفرستیم مثلا :
home/python_files/project1/"html.html" با مراحل بالا کل این مسیر استخراج میشود برای این که فقط نام فایل که بعد از آخرین / قرار دارد در خط ۱۷ توسط تابع rfind() آخرین / را پیدا کردیم و از انجا به بعد را در متغیر file_name ذخیره میکنیم و اگر هیچ / ایی نداشته باشیم تابع rfind() عدد ۱- را بر میگرداند که وقتی با یک جمع میشود عدد صفر میشود و این یعنی از ابتدای رشته ی قبلی که همان اسم فایل مد نظر بوده پس درصورت نبود / مشکلی به وجود نمی اید.
- در بین خطوط ۲۶ تا ۳۲ محتویات فایل را به صورت ۱۰۲۴ بایت خوانده و ارسال میکنیم اگر به انتهای فایل برسیم پیام end connection را ارسال میکنیم تا کلاینت متوجه اتمام فایل بشود.
توجه شود که time.sleep() برای این است که پیام end connection با پکت های ارسالی فایل یکی به حساب نیاید و در پکت جدا ارسال شود.
- در خطوط ۳۴ و ۳۶ استاتوس(status) های مورد نظر ذخیره شده اند .

سوالا های مربوط به wireshark

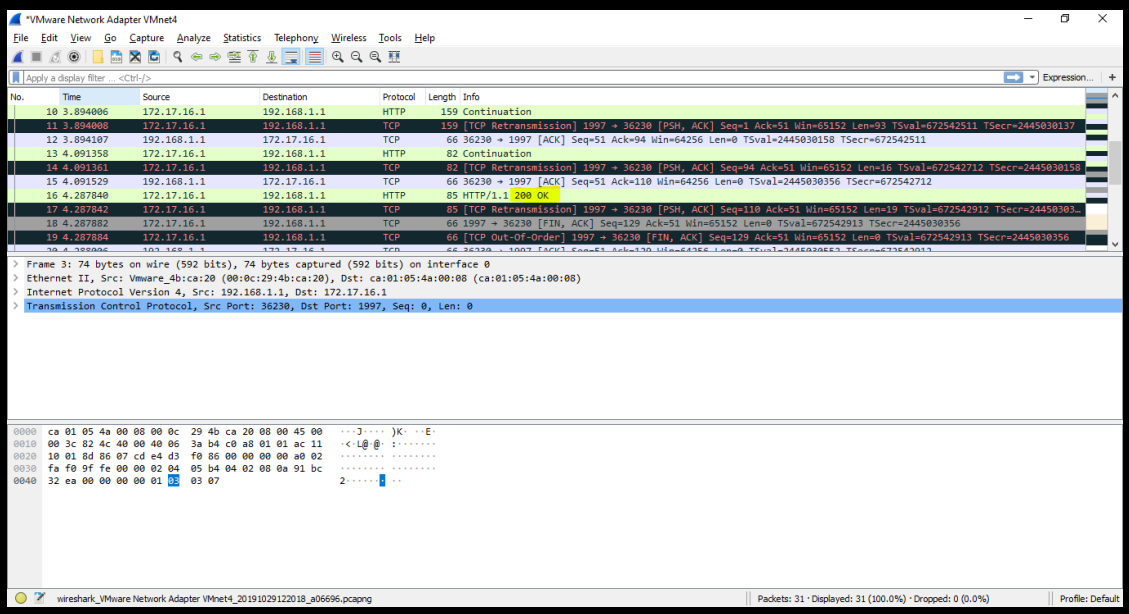
1)

ورژن http 1.1 ا ست. در تصویر زیر با رنگ زرد مشخص شده است.



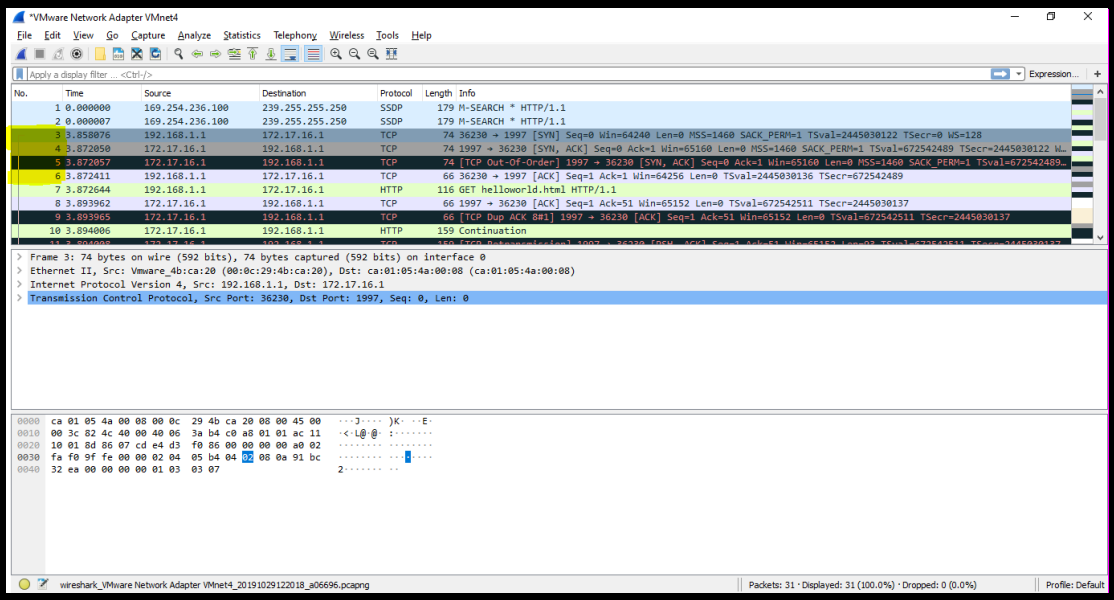
(2

در عکس زیر استاتوس ها با رنگ زد مشخص شده اند .استاتوس ها برای نشان دادن وضعیت درخواست ها کاربرد دارند. به صورت مثال ۲۰۰ یعنی این که با موفقیت ارسال شده.



(3

بله صورت گرفته در عکس زیر با رنگ زرد مشخص شده .



ابتدا کلاینت یک پیام با seq =0 فرستاده سپس سرور در جواب seq =0 و ACK=1 را ارسال کرده در مرحله بعد گویا به دلیل عدم دریافت در ترتیب درست سرور دوباره seq و ACK قبلی را ارسال کرده و در قسمت اخر کلاینت seq=1 و ACK=1 را ارسال کرده و اتصال برقرار شده.

سوال ۳)

در این سوال میخواهیم ده پینگ فرستاده و RTT هرکدام را محاسبه و در اخر میانگین آن ها را محاسبه کنیم.

کد سمت کلاینت را در تصویر زیر مشاهده میکنیم .

در خط ۱۳ یک ارایه برای تولید یک حلقه ی ده تایی برای تولید ده پینگ ایجاد کردم.

در خط های ۱۷ و ۱۸ متغییر هایی برای مقدار های مینیمم و ماکزیمم RTT ایجاد و مقدار دهی کردیم مقدار ها بر این اساس داده شده که بعد از اولین اجرای حلقه مقدار ها با مقدار های واقعی جایگزین شود و بعد از ان مقایسه صورت گیرد.

محاسبه RTT ها مشابه سوال یک است .بقیه کد ها کاملا واضح و ساده میباشند.

```
print("Usage: python UDPPingerClient\n <server ip address : 172.17.16.1> \r\n <server port no : 1997>")

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
clientSocket = socket(AF_INET, SOCK_DGRAM)

# To set waiting time of one second for reponse from server
#clientSocket.settimeout(1)

ten_ping = [1,2,3,4,5,6,7,8,9,10]
server_ip = "172.17.16.1"
serverPort = 1997
packetLost = 0
maxRtt = 0
minRtt = 100
RTT = 0
sum_of_RTTs = 0
client = socket(AF_INET, SOCK_DGRAM)
for counter in ten_ping:
    client.sendto(str(counter).encode("utf-8"), (server_ip, serverPort))
    timer1 = time.time()
    client.settimeout(2)# To set waiting time of one second for reponse from server

    try:
        receive = client.recv(1024)
        timer2 = time.time()
        if(int(receive) == counter):
            RTT = timer2 - timer1
            if RTT > maxRtt:
                maxRtt = RTT
            if RTT < minRtt:
                minRtt = RTT
            print("time for ping %d = %dms" % (counter, RTT * 1000))
        sum_of_RTTs = sum_of_RTTs+RTT
    except:
        packetLost = packetLost + 1
        print("ping %d has been lost"%counter)
print("packet loss =%d percent " %(packetLost*10))
mean_of_RTT = sum_of_RTTs/(10-packetLost)
print( "max RTT: %dms\nmin RTT: %dms\nmean RTT: %fms" % (maxRtt * 1000, minRtt * 1000, (mean_of_RTT) * 1000))
client.close()
```

در تصویر زیر کد سمت سرور را مشاهده میکنید:

UDP_pinger_server.py

```
#!/usr/bin/python3
from socket import *
server_port = 1997
server_ip = "172.17.16.1"
s = socket(AF_INET,SOCK_DGRAM)
s.bind((server_ip,server_port))
print("server is ready to be contacted ...\n")
while True:
    msg, client_ip=s.recvfrom(1024)
    s.sendto(msg, client_ip)
```

کاملا مشابه کد های قبلی است و چیز جدید ندارد.

خروجی کد را در زیر میبینیم :

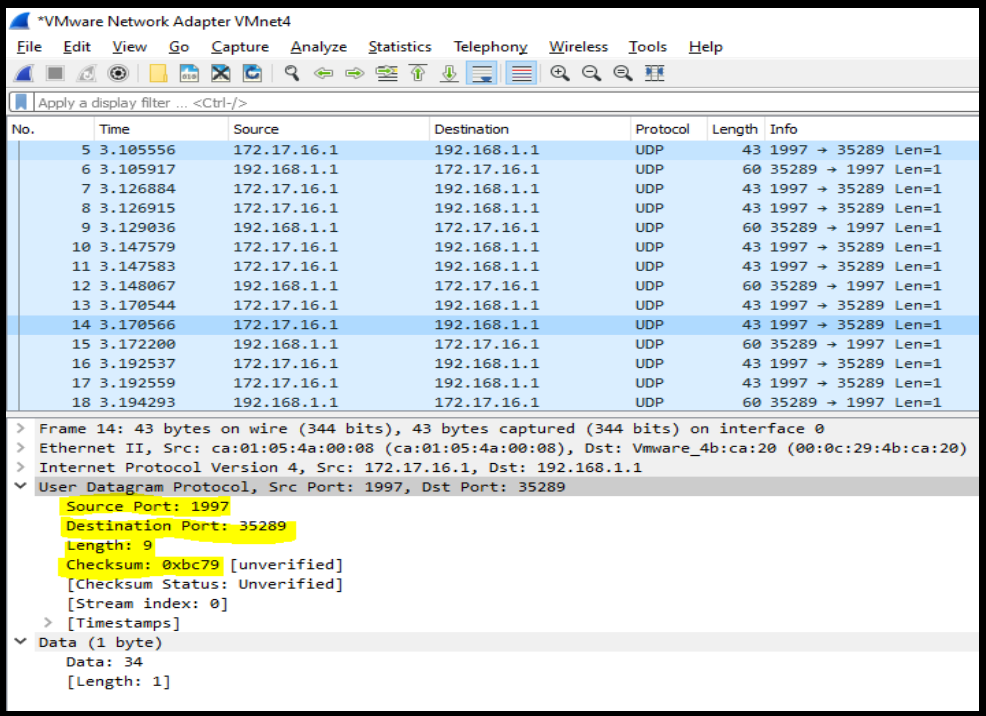
```
amiramri0200@ubuntu:~/python_files/project_1/part3$ python3 UDP_pinger_client.py

Usage: python UDPPingerClient
 <server ip address : 172.17.16.1>
 <server port no : 1997>
time for ping 1 = 18ms
time for ping 2 = 21ms
time for ping 3 = 21ms
time for ping 4 = 21ms
time for ping 5 = 20ms
time for ping 6 = 21ms
time for ping 7 = 20ms
time for ping 8 = 21ms
time for ping 9 = 20ms
time for ping 10 = 21ms
packet loss =0 percent
max RTT: 21ms
min RTT: 18ms
mean RTT: 21.033144ms
amiramri0200@ubuntu:~/python_files/project_1/part3$
```

سوالا های مربوط به wireshark

1)

چهار header وجود دارد با نام های checksum و destination port و source port و length که در تصویر زیر با رنگ زرد مشخص شده .



۲)

طول هدر و داده های داخل UDP را مشخص به بایت مشخص میکند .کمترین این مقدار ۸ باید میباشد.

سوال ۴

هدف ایجاد یک میل کلاینت ساده است .

در تصویر زیر کد را مشاهده میکنید:

```
mail_client.py                                                    buffers
 1 #!/usr/bin/python3
 2 import smtplib, imaplib, email
 3
 4 username = input('user name: ')
 5 password = input('password: ')
 6
 7 server = smtplib.SMTP('smtp.gmail.com', 587)
 8 server.ehlo()
 9 server.starttls()
10 server.login(username, password)
11
12 subject = input('subject: ')
13 message = input('message:\n')
14 recipient = input('recipient: ')
15 header = 'to:' + recipient + '\n' + 'from:' + username + '\n' + 'subject:' +
16 content = header + '\n' + message + '\n'
17 server.sendmail(username, recipient, content)
18 print("your email has been sent successfuly")
19
~
~
NORMAL  mail_client.py          pyt…   5% ☰   1/19 ㏑:  1  ☲[1]trai…
"mail_client.py" 19L, 538C
```

کتابخانه smtplib را برا کار با mail را اضافه میکنیم.

در خط ۷ به gmail و پورت مربوط به ان وصل میشویم

از کدینگ ttls استفاده کردیم (خط ۹)

در ادامه با گرفتن موضوع و محتوای پیام آن در یک متغییر قرار داده و سپس ارسال کرده ایم.