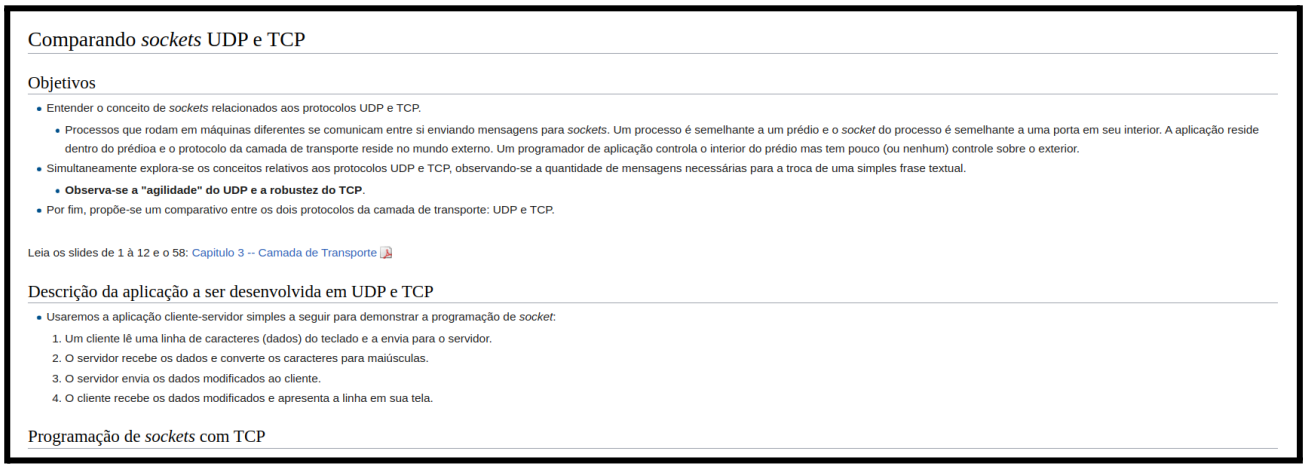
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES – CÂMPUS SÃO JOSÉ

RELATÓRIO TÉCNICO

SOCKETS TCP E UDP

Arthur Cadore Matuella Barcella

TAREFA:



Comparando *sockets* UDP e TCP

Objetivos

- Entender o conceito de *sockets* relacionados aos protocolos UDP e TCP.
  - Processos que rodam em máquinas diferentes se comunicam entre si enviando mensagens para *sockets*. Um processo é semelhante a um prédio e o *socket* do processo é semelhante a uma porta em seu interior. A aplicação reside dentro do prédioa e o protocolo da camada de transporte reside no mundo externo. Um programador de aplicação controla o interior do prédio mas tem pouco (ou nenhum) controle sobre o exterior.
- Simultaneamente explora-se os conceitos relativos aos protocolos UDP e TCP, observando-se a quantidade de mensagens necessárias para a troca de uma simples frase textual.
  - **Observa-se a "agilidade" do UDP e a robustez do TCP**.
- Por fim, propõe-se um comparativo entre os dois protocolos da camada de transporte: UDP e TCP.

Leia os slides de 1 à 12 e o 58: Capitulo 3 -- Camada de Transporte

Descrição da aplicação a ser desenvolvida em UDP e TCP

- Usaremos a aplicação cliente-servidor simples a seguir para demonstrar a programação de *socket*:
  1. Um cliente lê uma linha de caracteres (dados) do teclado e a envia para o servidor.
  2. O servidor recebe os dados e converte os caracteres para maiúsculas.
  3. O servidor envia os dados modificados ao cliente.
  4. O cliente recebe os dados modificados e apresenta a linha em sua tela.

Programação de *sockets* com TCP

Capturas da configuração inicial utilizando um notebook com ubuntu 22.04 (lembrando que: o código python precisou ser alterado para a versão do python 3).

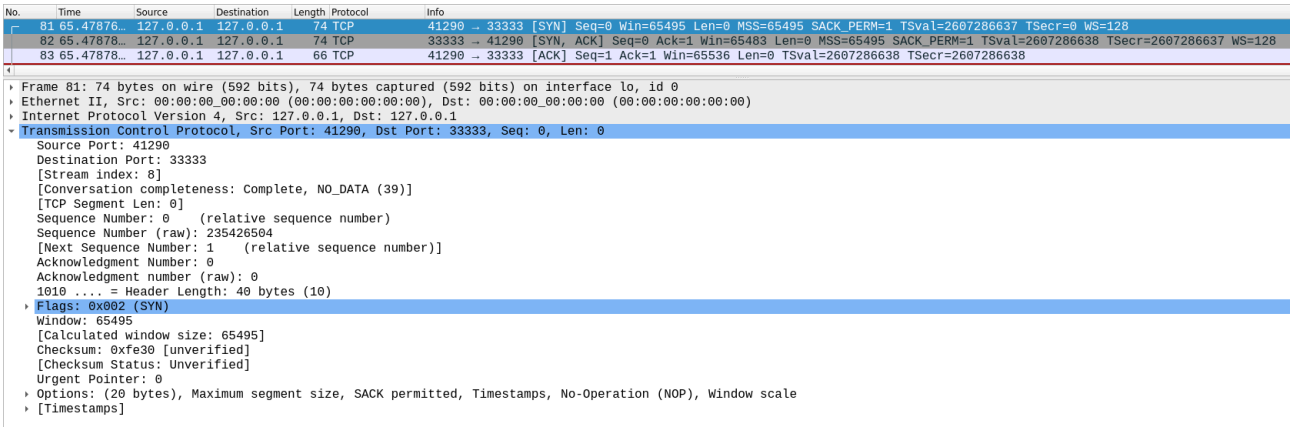
```
cadore@cadore-ac:~/IFSC/redes$ cat TCPServer.py
from socket import *
serverPort = 33333
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('',serverPort))
#Escuta as requisicoes do TCP do cliente. Numero maximo de conexoes em fila = 1
serverSocket.listen(1)
print ('O servidor esta pronto')
while 1:
        #Quando o cliente bate a essa porta, o programa chama o metodo accept() para serverSocket,
        #que cria um novo socket no servidor, chamado connectionSocket, dedicado a esse cliente
        #especifico. Cliente e servidor, entao, completam a apresentacaoo, criando uma conexao TCP
        #entre o clientSocket do cliente e o connectionSocket do servidor.
        connectionSocket, addr = serverSocket.accept()
        message = connectionSocket.recv(1024)
        print (message)
        messageMaiuscula = message.upper()
        connectionSocket.send(messageMaiuscula)
        connectionSocket.close()
```

```
cadore@cadore-ac:~$ nmap -p 3333 127.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-10 10:17 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000068s latency).

PORT     STATE  SERVICE
3333/tcp closed dec-notes

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
cadore@cadore-ac:~$ nmap -p 33333 127.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-10 10:17 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000069s latency).

PORT      STATE SERVICE
33333/tcp open  dgi-serv

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
```

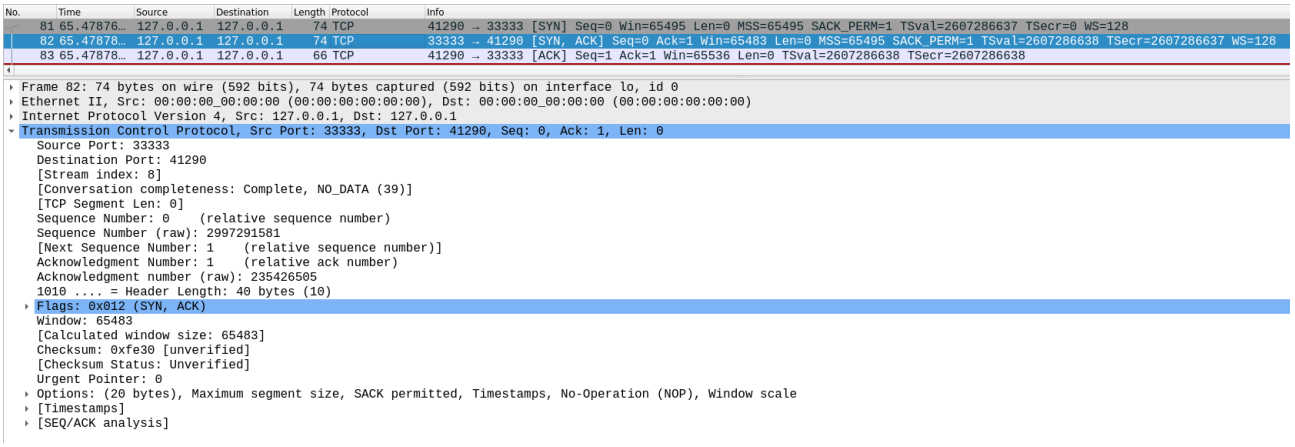
1) **As três primeiras mensagens trocadas apresentam a camada de aplicação, sim ou não? Explique. O que elas significam?**

Não apresentam, apenas mensagens até a camada de transporte. Servem para a troca de mensagens inicial entre as partes, para negociar parâmetros de sessão e iniciar a sessão.

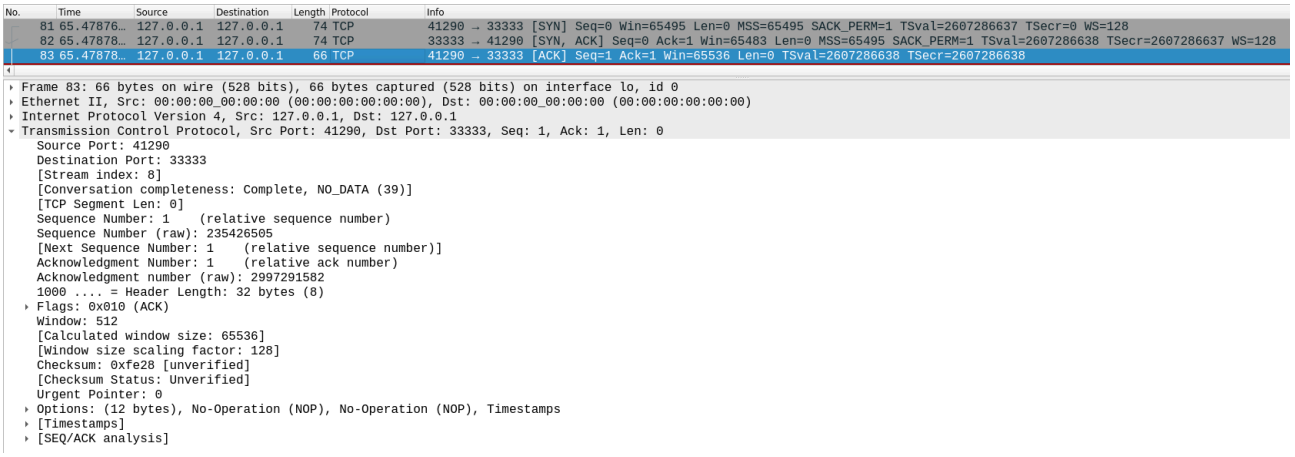
Primeira mensagem SYN:



Segunda mensagem SYN, ACK:

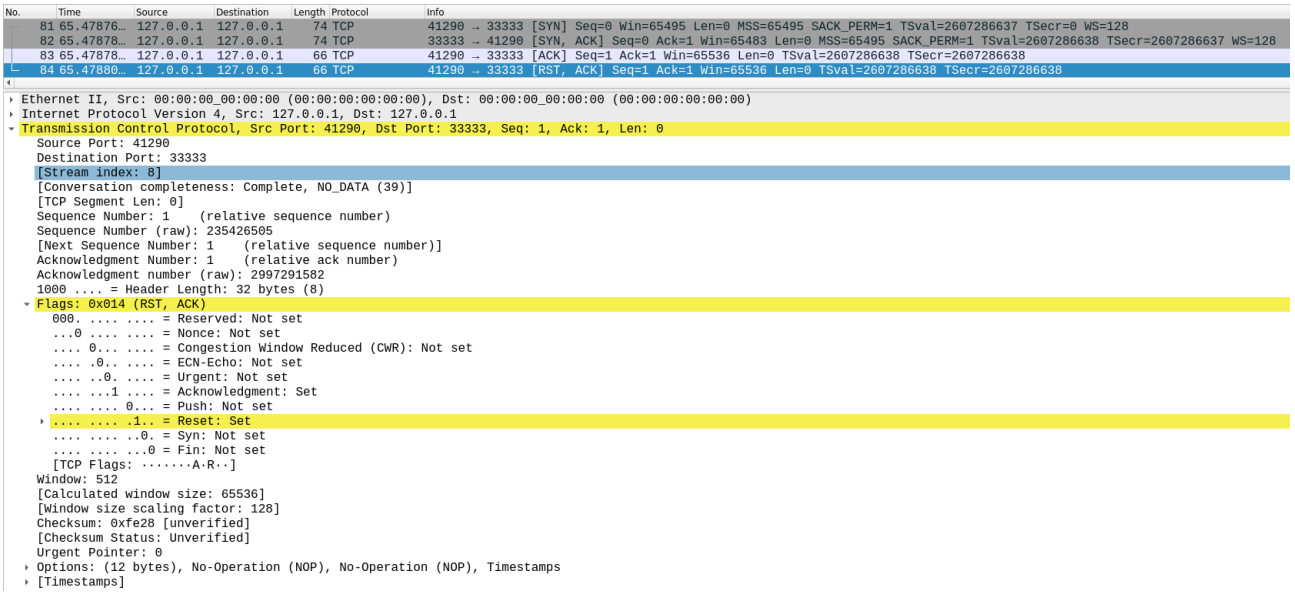


Terceira mensagem ACK:



2) **Em qual mensagem (número) é que a frase escrita é enviada ao servidor?**

A partir da quarta mensagem, até o momento da finalização da conexão, todas as demais mensagens são encaminhadas entre servidor e cliente para troca de dados úteis.



3))A mensagem seguinte (quinta) apresenta a camada de aplicação? Clique na camada TCP no Wireshark e observe o campo Flags: 0x010 (ACK). O que você acha que isso significa?

R: Significa que junto da mensagem que voltou ao cliente, foi encaminhado também um ACK como "carona" no pacote.

4) Qual o conteúdo da mensagem seguinte (sexta)? E da sétima? Explique.

R: Durante a sexta e sétima mensagem, o cliente solicitou uma página ao servidor através de HTTP e o servidor encaminha um ACK através do TCP para informar ao cliente que a solicitação foi recebida.

5) Qual é o protocolo da camada de transporte nessa troca de mensagens?

R: TCP

Quais são os números de porta e os IPs utilizados?

R: Origem 41290

Destino 33333

6) Quais foram os números de sequência utilizados em todas as mensagens?

Iniciou-se em 0 (convertido pelo wireshark) e finalizou em 0, pois o outro dispositivo não transmitiu dados, apenas recebeu do dispositivo inicial.

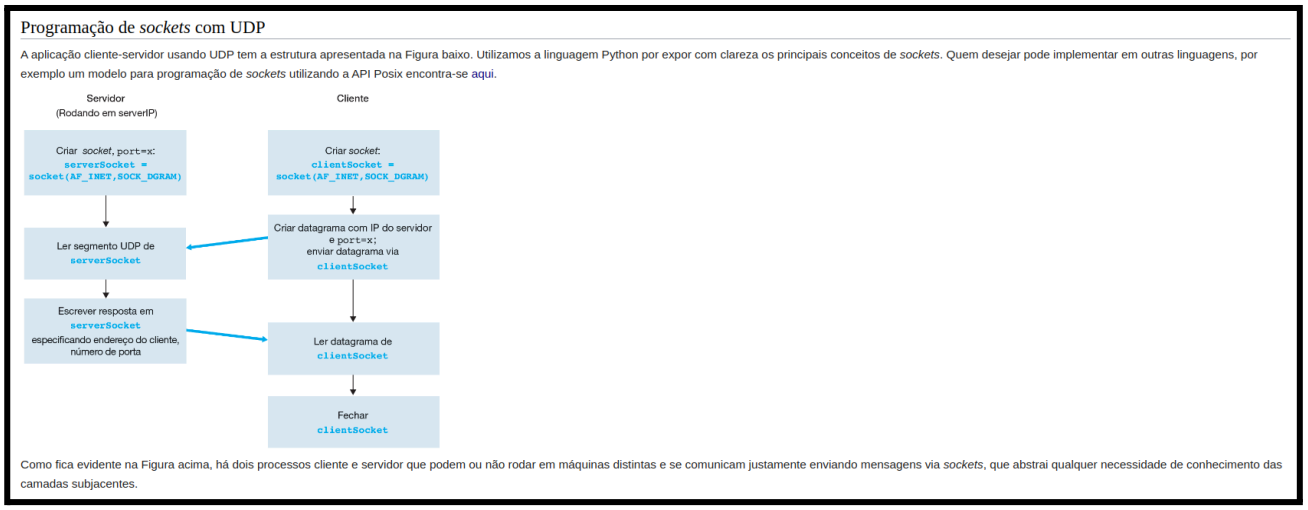
Qual o número identificador de protocolo TCP no pacote IP? Dica: na janela central abra o campo Internet Protocol e procure a string Protocol.

Campo "Protocol" sendo que seu valor é: 6.

Quantos sockets foram abertos no servidor com um cliente "conectado"? E com dois clientes?

Com um cliente foi aberto apenas um socket, com dois clientes, foram abertos dois sockets.

PARTE 2 (SOCKETS UDP):

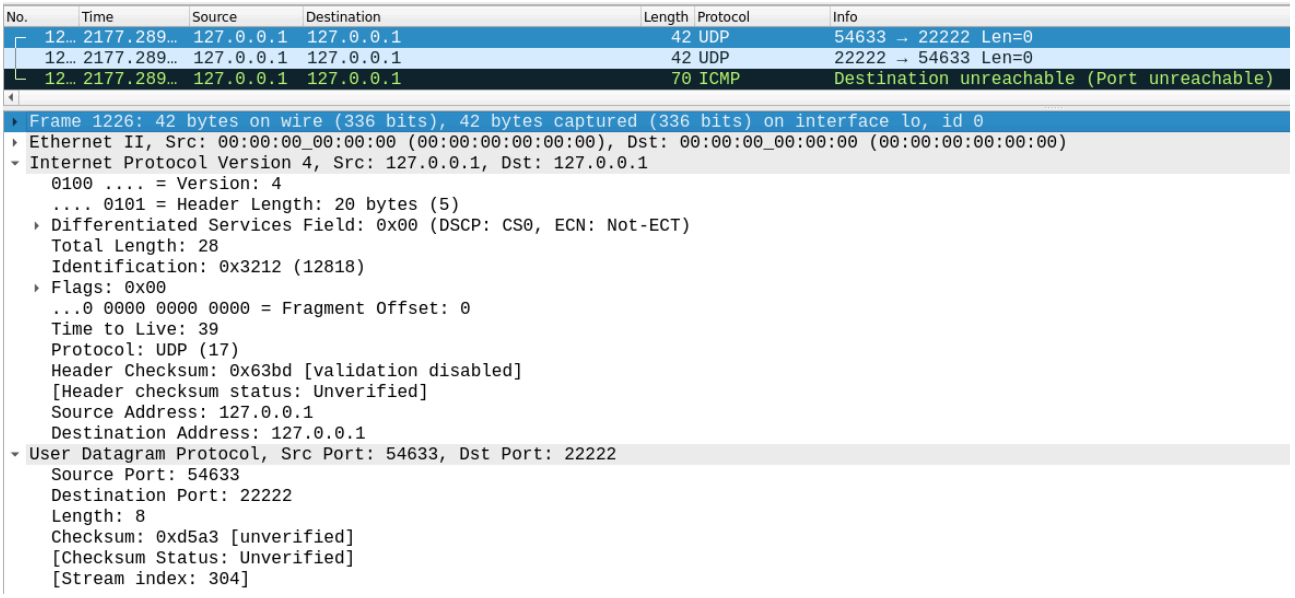


Verificando as conexões ativas no netstat, veja que a porta 22222 está aberta para o protocolo UDP.

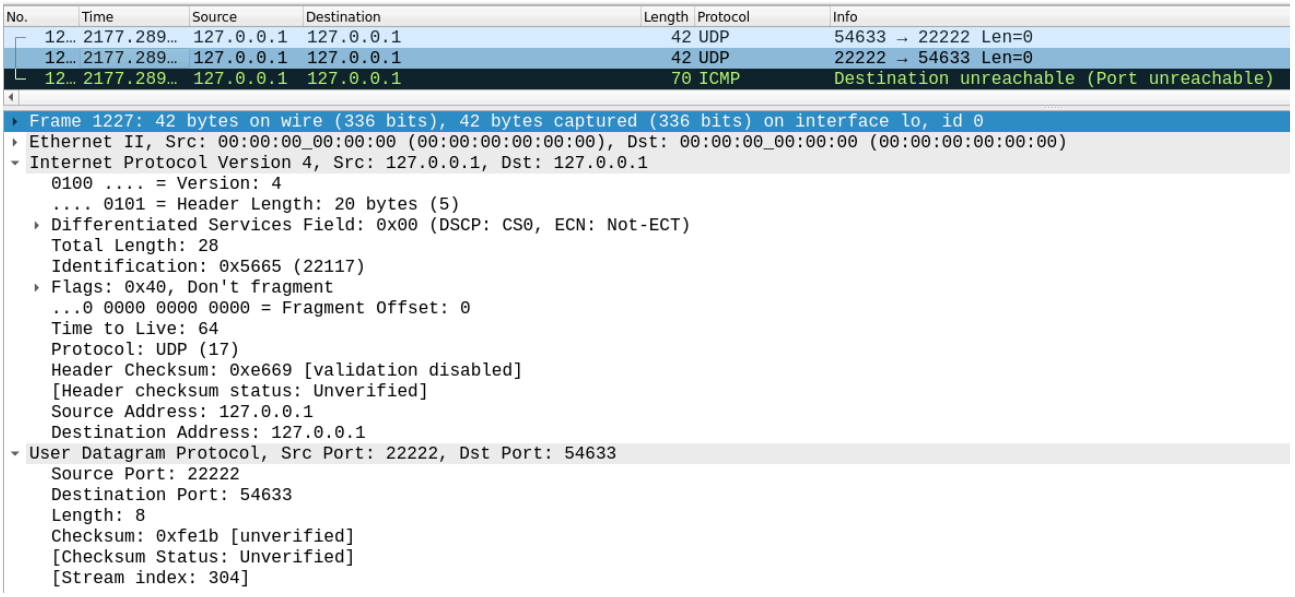
```
cadore@cadore-ac:~$ netstat -eau
Conexões Internet Ativas (servidores e estabelecidas)
Proto Recv-Q Send-Q Endereço Local          Endereço Remoto        Estado       Usuário Inode
udp        0      0 localhost:domain        0.0.0.0:*                           systemd-resolve 23551
udp        0      0 cadore-ac:bootpc        191.36.8.66:bootps     ESTABELECIDA root       3329634
udp        0      0 0.0.0.0:631             0.0.0.0:*                           root       1632225
udp        0      0 224.0.0.251:mdns        0.0.0.0:*                           cadore     3331328
udp        0      0 224.0.0.251:mdns        0.0.0.0:*                           cadore     2352414
udp        0      0 224.0.0.251:mdns        0.0.0.0:*                           cadore     95216
udp        0      0 224.0.0.251:mdns        0.0.0.0:*                           cadore     95214
udp        0      0 0.0.0.0:mdns            0.0.0.0:*                           avahi      26863
udp        0      0 cadore-ac:38318         52.97.26.34:https      ESTABELECIDA cadore     3330453
udp        0      0 0.0.0.0:22222           0.0.0.0:*                           cadore     3444663
udp        0      0 0.0.0.0:35145           0.0.0.0:*                           avahi      26865
udp6       0      0 cadore-ac:58104         2800:3f0:4001:828:https ESTABELECIDA cadore     3442536
udp6       0      0 [::]:mdns               [::]:*                              avahi      26864
udp6       0      0 [::]:42805              [::]:*                              avahi      26866
udp6       0      0 cadore-ac:55418         2800:3f0:4001:829:https ESTABELECIDA cadore     3446903
udp6       0      0 cadore-ac:47241         2800:3f0:4001:82e:https ESTABELECIDA cadore     3403292
udp6       0      0 cadore-ac:47281         2800:3f0:4001:82e:https ESTABELECIDA cadore     3409182
udp6       0      0 cadore-ac:60098         2800:3f0:4001:832:https ESTABELECIDA cadore     3444694
udp6       0      0 cadore-ac:48772         2800:3f0:4001:80f:https ESTABELECIDA cadore     3447886
udp6       0      0 cadore-ac:40662         2800:3f0:4001:81f:https ESTABELECIDA cadore     3442652
```

Feita uma captura no wireshark e testado o programa para socket UDP.

Note que não há pacotes de estabelecimento de conexão antes da passagem de dados, os três primeiros pacotes já são responsáveis por encaminhar dados UDP.



O Segundo pacote responde ao solicitante com dados de retorno, mas da mesma maneira como visto na mensagem anterior, não houve estabelecimento de conexão.



PERGUNTAS baseadas na captura:

1) **Em algum momento foi identificado algum procedimento para estabelecimento de conexão?**

Para o protocolo UDP não.

2) **Em algum campo do UDP existe numeração de mensagens?**

Não, o protocolo apenas encaminha as mensagens, sem numeração.

3) **Qual o número identificador de protocolo UDP no pacote IP? Dica: na janela central abra o campo Internet Protocol e procure a string Protocol.**

Numero 17.

4) **Qual é o checksum no pacote (datagrama) UDP? Qual é o formato apresentado? Quantos bits ele possui?**

É formado pelo campo "checksum", ele possui 4 caracteres hexadecimais, portanto 16bits.

5) **É possível capturar toda a troca de mensagens e inclusive capturar o texto passado do cliente para o servidor?**

Sim, pois ele não foi cifrado antes do encaminhamento da mensagem.

Comparativo entre TCP e UDP:

1) **Se a mensagem digitada for teste, do cliente para o servidor deve aparecer o campo Data: 7465737465 e a resposta do servidor deve aparecer Data: 5445535445. O que significa isso? Dica, olhe na internet o código ASCII.**

Significa que o conteúdo da mensagem do pacote é 5445535445.

2) **Qual foi a sequência numérica do campo Data em seu teste? Qual o significado?**

Significa que o pacote é a sequência X da conexão.

3) **Qual é o protocolo da camada de transporte nessa troca de mensagens?**

TCP devido a cifra utilizada, sequência de conexão, etc.

4) **Quantos sockets foram abertos no servidor com um cliente "conectado"? E com dois clientes?**

Apenas 1, no caso de ter 2 clientes, dois sockets seriam abertos.