Trabalho 1 Laboratório de Redes

Nome: Gabriel Tasca Villa
Matrícula: 18106131

Funcionamento TCP:
Rodando server e depois client temos a seguinte mensagem:

```
gabriel@tasca:~/Desktop/Faculdade/Lab Redes/t1_labredes/Socket-TCP$ python3 client.py
Digite 'm' para enviar uma mensagem ou 'f' para enviar um arquivo (ou 'exit' para sair):
```

Mensagem digitada e resposta do server:

```
Digite a mensagem: Hello World!
Resposta do servidor: Mensagem recebida com sucesso!
```

Conexão com o server feita e mensagem do client logada:

```
Aguardando conexão...
Conexão de: ('127.0.0.1', 45100)
Mensagem recebida do cliente ('127.0.0.1', 45100): Hello World!
```

Envio da mensagem para o Server:

```
▸ Frame 784: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 45100, Dst Port: 5000, Seq: 1, Ack: 1, Len: 15
▾ Data (15 bytes)
    Data: 4d534748656c6c6f20576f726c6421
    [Length: 15]
```

Envio da mensagem de retorno para o Client:

```
▸ Frame 786: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 5000, Dst Port: 45100, Seq: 1, Ack: 16, Len: 30
▾ Data (30 bytes)
    Data: 4d656e736167656d207265636562696120636f6d207375636573736f21
    [Length: 30]
```

Envio de arquivo:

- Obs: o arquivo deve estar na pasta do servidor.

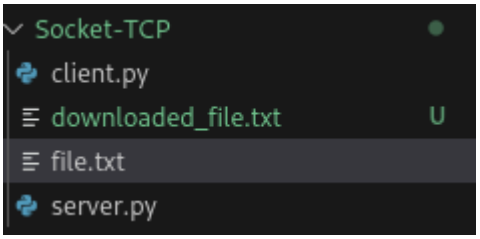
Cliente mandando o nome do arquivo desejado e resposta do servidor depois que arquivo foi baixado

```
Resposta do servidor: Mensagem recebida com sucesso!
Digite 'm' para enviar uma mensagem ou 'f' para enviar um arquivo (ou 'exit' para sair): f
Digite o nome do arquivo a ser enviado: file.txt
Arquivo "file.txt" baixado com sucesso como "downloaded_file.txt"
```

Solicitação do arquivo logado no servidor.

```
Arquivo solicitado pelo cliente ('127.0.0.1', 45100): file.txt
```

Arquivo baixado na pasta do cliente




```
▸ Frame 13288: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 5000, Dst Port: 45100, Seq: 31, Ack: 28, Len: 12
▾ Data (12 bytes)
    Data: 48656c6c6f20576f726c6421
    [Length: 12]
```

```
▸ Frame 13287: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 45100, Dst Port: 5000, Seq: 16, Ack: 31, Len: 12
▾ Data (12 bytes)
    Data: 46494c4566696c652e747874
    [Length: 12]
```

Funcionamento UDP:

Funcionamento parecido com o TCP

```
Digite 'm' para enviar uma mensagem ou 'f' para enviar um arquivo: m
Digite a mensagem: Hello World!
Digite 'm' para enviar uma mensagem ou 'f' para enviar um arquivo: f
Digite o nome do arquivo a ser enviado: file.txt
Digite 'm' para enviar uma mensagem ou 'f' para enviar um arquivo:
```

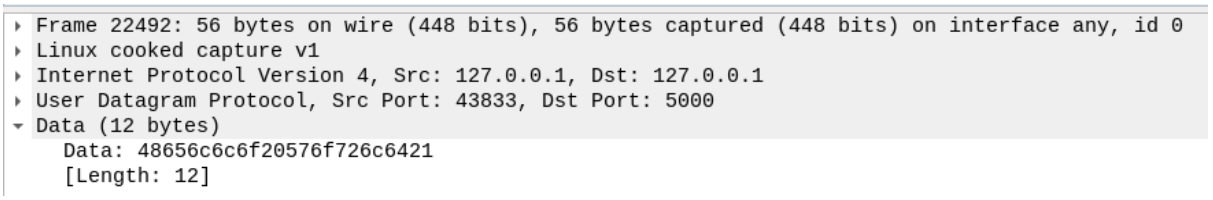
Logs de mensagens recebidas pelo servidor de solicitações diferentes:

```
Mensagem recebida do cliente ('127.0.0.1', 43833): Hello World!
Mensagem recebida do cliente ('127.0.0.1', 49905): Hello world!
```

Depois que mensagem chegou:

```
▸ Frame 22497: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ User Datagram Protocol, Src Port: 49905, Dst Port: 5000
▾ Data (12 bytes)
    Data: 48656c6c6f20776f726c6421
    [Length: 12]
```

Aqui seria o envio do arquivo:



Perguntas:

1) Porta de origem - 45100 Porta de destino - 5000 para para requisições cliente -> servidor
   Porta de origem - 5000 Porta de destino - 45100 para para requisições servidor -> cliente
2) Sim, mínimo do socket TCP foi 80 bytes e o UDP foi 50 bytes
3) Não. As aplicações não apresentaram diferença.